THE UNIVERSITY OF ALBERTA

LOGO PROGRAMMING BUGS AND DEBUGGING STRATEGIES OF GRADE SIX STUDENTS

by

© GLORIA M. CATHCART

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND RESEARCH

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE

OF MASTER OF EDUCATION

DEPARTMENT OF ELEMENTARY EDUCATION

EDMONTON, ALBERTA

FALL, 1985

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Gloria M. Cathcart

TITLE OF THESIS: "Logo Programming Bugs and Debugging Strategies of Grade Six Students"

DEGREE: Master of Education

YEAR THIS DEGREE GRANTED: 1985

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

Mrs. Gloria Cathcart

Permanent Address:

11511 - 35A Avenue

Edmonton, Alberta

T6J 0A9

Date: June 14, 1985

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled "Logo Programming Bugs and Debugging Strategies of Grade Six Students" submitted by Gloria M. Cathcart in partial fulfilment of the requirements for the degree of Master of Education.

Joan Worth

Supervisor

Daiyo Sawada

Louis E. Keeran

Date: June 14, 1985

ABSTRACT

This study invesitgated the Logo programming of 20 grade 6 students in 3 Edmonton schools. The major purposes of the study were to identify and classify the bugs present in the Logo work of students, identify the strategies that students used to debug Logo programs, and describe the attitudes of students toward bugs.

Data were collected through the observation of individual students and took place during January and February, 1985. Students chosen by the teachers as representative of their classes in regards to Logo programming exhibited a fairly wide range of programming ability. Actual programming assignments observed were those given by the teacher as a part of the ongoing program.

Each of the 20 students involved in the study was observed an average of 4 times, with an average total observation time of 2 hours per student. Each student was also involved in a preliminary acquainting session and culmination questionnaire and tape recorded interview sessions. During the observations, extensive notes were taken, disk files kept, and tape recordings made where possible. The notes taken included the actual input students were entering at the computer keyboard, the comments made and actions observed, and notations about attitudes evident. Log notes were also kept.

A total of 528 bugs were recorded and classified into 8 categories. The categories (showing frequency) of bugs were dimension (31%), lateralization (18%), omission (14%), orientation (13%), locating origin (9%), syntax (5%), excess (3%), and miscellaneous bugs (7%). The 1206 identified strategies, grouped under 14 headings, were

guess and check (41%), observe and change (12%), immediate mode (9%), clear screen and start again (8%), physical referent (7%), delete and start again (5%), follow pattern (4%), search listing (3%), error messages (2%), simplify (1.5%), seek information (1%), change plan of approach (1%), abandon or change task (0.5%), and miscellaneous strategies (5%).

Some bugs were easily corrected, taking only one attempt, while others were difficult and time consuming, requiring several attempts and the use of several strategies. Lateralization bugs were the easiest to debug, while locating origin bugs were the hardest. Those students considered to be more advanced programmers were also those who used a wider variety of debugging strategies.

Assignments given to students were fairly open ended and resulted in a wide range of completed projects. Little evidence of detailed planning prior to keyboard work, or looking back to refine a procedure or make it more efficient was noted. Except for some note taking, a few diagrams drawn, and the use of a physical referent, students worked without the use of concrete aids.

Attitudes, as indicated by the responses to the adapted Nyberg and Clarke (1979) attitude scales and the observations and interviews by the researcher, ranged from positive to very positive. Although some students indicated they did not like bugs, tasks were handled with much perserverance and little sign of frustration. Abandonment of task occurred only 3 times in 81 observation sessions, indicating a strong commitment to the completion of tasks and the challenge of debugging.

ACKNOWLEDGEMENTS

Appreciation is extended to all those who provided support and encouragement in the preparation of this thesis. In particular I wish to thank:

Dr. Joan Worth, thesis supervisor, whose advice, guidance, and encouragement were readily available and greatly appreciated.

Dr. Daiyo Sawada and Dr. Tom Kieren, members of my examining committee, and Dr. Don Massey, who offered constructive suggestions and support as the work progressed.

The teachers and students in the three schools who served as subjects for this study and those in the pilot study school, whose professional cooperation and generous assistance made this study possible. Assurance of anonymity precludes a more specific acknowledgement of their greatly appreciated contribution.

My husband, George, who so willingly took on extra duties on the homefront, provided encouragement and advice, and displayed unending patience.

My children, Cori Lyn and Curtis, for assisting and coping with a somewhat preoccupied mother.

TABLE OF CONTENTS

CHAPTER		PAGE
I.	THE PROBLEM AND THE NATURE OF THE INVESTIGATION . . .	1
	Introduction	1
	The Purpose of the Study	2
	Review of the Related Literature	3
	Types of Bugs and Debugging Strategies - Not Logo Specific	3
	Types of Bugs and Debugging Strategies in Logo .	6
	Attitude of Students in Relation to Debugging .	9
	Summary	12
	Definitions	12
	Limitations	14
	Delimitations	14
	Significance	14
	Outline of the Study	15
II.	RESEARCH DESIGN AND SETTING	16
	Design of the Investigation	16
	Sample	17
	Informal Pilot Study of Procedures	17
	Observation Sheet	17
	The Attitude Scale	18
	Procedure	19
	Phase One	19
	Phase Two	21
	Phase Three	22

CHAPTER		PAGE
II.	Time Frame	22
	Adjustments to the Design	24
	Deletion of One Set of Observations	24
	Interventions by the Researcher	24
	Settings	25
	School A	25
	School B	26
	School C	26
	The Students in the Study	28
	Programming Assignments	30
	Assignments Provided by the Teacher	30
	Assignments Interpreted by the Students	33
	School A	33
	School B	34
	School C	36
	Adapting Assignments	38
	Summary	45
III.	RESULTS OF THE INVESTIGATION	46
	Data Analysis Procedures	46
	Analyzing the Observation Sheets	46
	Reliability Check	47
	Attitude Questionnaires	48
	Bugs Observed in Students' Work	49
	Dimension Bugs	52

CHAPTER		PAGE
III.	Orientation Bugs	52
	Lateralization Bugs	53
	Locating Origin Bugs	54
	Omission Bugs	55
	Excess Information Bugs	55
	Syntax Bugs	56
	Miscellaneous Bugs	56
	Debugging Strategies Observed in Students' Work	56
	Immediate Mode	60
	Observe and Change, Add or Delete	61
	Guess and Check	61
	Playing Computer or Playing Turtle	62
	Clear Screen and Start Again	63
	Delete and Start Again	63
	Simplify	64
	Seek Information	64
	Following a Pattern	65
	Error Messages	65
	Changing the Plan of Attack	65
	Abandon or Change Task	65
	Miscellaneous Strategies	66
	How Strategies Relate to Bugs	67
	Attitude	71
	Attitude Scale - Debugging Computer Programs	72

CHAPTER		PAGE
III.	Final Interviews	75
	How Do You Feel About Bugs?	77
	Learning More About Logo	79
	Bugs and Causes	80
	Strategies	82
	Other Considerations	84
	Mode of Development	85
	Procedures and Subprocedures	86
	Preventative Debugging	89
	Refining, Expanding, and Exploring	89
	Off-Computer Debugging	90
	Number of Tries Per Bug	91
	Time Verses Accomplishment	91
	Bug Serendipity	95
	Bugs Verses Ability	95
	Styles of Programming Exhibited by Individuals	95
	Other Difficulties Students Encountered	97
	Summary	98
IV.	SUMMARY, FINDINGS AND IMPLICATIONS, CONCLUSIONS AND RECOMMENDATIONS	99
	Summary	99
	Discussion and Implications of Findings	100
	Bugs	100
	Debugging Strategies	103

CHAPTER PAGE

IV. Attitude 107
Conclusions 108
Recommendations for Research 109
Recommendations for Teaching 111

REFERENCES 113

APPENDIX 1: Observation Sheet 117
APPENDIX 2: Debugging Computer Programs 119
APPENDIX 3: Student Interview 121
APPENDIX 4: Letter to Parents 124
APPENDIX 5: Complete Observation Schedule 126
APPENDIX 6: School C Assignments 130
APPENDIX 7: Sample Completed Observation Sheets and Log Notes 134
APPENDIX 8: Bug Specimen 147
Dimension Bugs 148
Orientation Bugs 151
Lateralaization Bugs 152
Origin Bugs 153
Omission Bugs 158
Excess Information Bugs 160
Syntax Errors 161
Miscellaneous Bugs 162

PAGE

APPENDIX 9: Debugging Strategy Specimen 166

Immediate Mode 167

Observe and Change 170

Guess and Check 171

Playing Computer or Playing Turtle 173

Clear Screen and Start Again 174

Delete and Start Again 176

Simplify 177

Seeks Information 180

Following a Pattern 183

Error Messages 184

Changing the Plan of Attack 187

Abandon or Change Task 188

Other Strategies 189

Mathematical Calculations 189

Drawing Diagrams 190

Note Taking 192

Abandoned Procedures 193

Change Order of Procedures 194

Using <CONTROL> G 194

Using Variables 194

Using PRINT POS 195

Experimenting 196

Multiple Strategies 196

PAGE

APPENDIX 10: Observation and Log Notes Illustrating Student Interest 199

Observation Notes on Cindy 200

Observation and Log Notes on Mark 201

LIST OF TABLES

TABLE		PAGE
1.	Observation Schedule	23
2.	IQ and Mathematics Scores	29
3.	Reliability Check	48
4.	Bugs Identified	51
5.	Debugging Strategies Identified	59
6.	Relationship Between Type of Bugs and Debugging Strategies Used to Eradicate Each Bug	69
7.	Attitude Scores - Total Score	73
8.	Attitude Scores - Subscales	73
9.	Strategies Students Say They Use	83
10.	Immediate Mode Verses Procedural Mode for Program Development	85

LIST OF FIGURES

FIGURE		PAGE
1.	Valentine Project by Laura	40
2.	Valentine Design by Mike	41
3.	Tessellation of Squares by Vicki	42
4.	Tessellation of Hexagons by Ray	43
5.	Serendipidy Designs from Bugs	44
6.	Trevor's Hexagon Tessellation	87
7.	Connie's Work - Three Procedures - 30 Minutes	92
8.	Ray's Work - A Castle - 35 Minutes	93
9.	Carol's Work - Symmetrical Squares - 40 Minutes	94
10.	Ted's Tessellation	157
11.	Don's Graphic	165
12.	Murray's Calculations	190
13.	Laura's Notes and Diagrams	191
14.	Sandy's Graph Paper Drawing	192
15.	Ted's Variable Input Procedure	195

CHAPTER I

THE PROBLEM AND THE NATURE OF THE INVESTIGATION

Introduction

"Now I know what it means to learn from my mistakes!"

- Sara, 8 years old -

The impact of the computer as a tool in our society has resulted in the inclusion of computers in many of our schools. The Government of Alberta (1983) has prepared the Curriculum Guide for Elementary Computer Literacy and many schools have implemented a computer literacy program. The computer is also being used extensively for computer assisted instruction. Logo, a computer language, has become increasingly popular in elementary schools. This is evidenced by the appearance of Logo programs at the local student computer fair, the scheduling and conducting of Logo teacher inservice sessions, the proportion of Logo sessions at recently scheduled conferences, and the large number of Logo articles and books that are currently being published.

Logo has a number of characteristics, such as its graphics qualities, simple commands, and user control, which have been influential in its impact on the elementary school classroom. Logo is referred to as a "no threshold, no ceiling" language. It is very easy for young children to become involved in Logo programming, but Logo also provides challenge and stimulation to the advanced programmer.

Programming in Logo may be likened to composing in English, although the final product may be graphic rather than textual, since the graphics part of Logo is presently more popular in our elementary schools. Even as children have errors in their creative writing, they have errors in their Logo computer programs. Editing a Logo program to remove the errors, or bugs as they are called in computer terminology, is a very important step in programming, and often takes place right along with the program development. Students often write a section or module of a program, test it, and then debug. This editing, or debugging, requires the implementation of a number of strategies, since a variety of bugs occur. Because this is an important and powerful aspect of computer work with children, it has become the focus of this study. The opening statement by Sara stimulated a desire to know more about the errors children make and how they go about correcting them.

The Purpose of the Study

The central objective of this exploratory study was to examine the nature of students' debugging of their computer programs written in Logo.

More specifically the purposes were:

1. to identify and classify the bugs present in the work of students,
2. to identify the strategies that students use to debug Logo programs, and
3. to describe the attitudes of students toward bugs.

Review of the Related Literature

The discussion of the related literature is divided into three categories. Since there has been little research in the area of bugs students encounter and the debugging strategies they use when programming in Logo, the first category deals with the identification of types of bugs and debugging strategies discussed in computer literature not specific to Logo. The second category reviews the literature relevant to bugs and debugging strategies specific to Logo. The third category deals with the attitude of students in relation to debugging in Logo.

Types of Bugs and Debugging Strategies - Not Logo Specific

Debugging, which is the focus of this study, is one of the activities of a programmer included in Shneiderman's (1980) list of programming tasks. He describes debugging as a "difficult task, which is best avoided" (p. 46). A bug-free program is most often the result of debugging and seldom the result of a program without bugs in the initial writing. Bruce (1980) states that "experience brings wisdom, but it only rarely brings perfection; oldtimers who have been programming since the days of the first UNIVAC still manage to commit errors in their programs and wind up on a bug hunt" (p. xiv).

Most writers classify bugs according to type. Shneiderman (1980) uses a syntactic/semantic model. He refers to syntactic errors as those related to the syntax of the programming language, while semantic errors are those of design or composition. Since syntactic errors are detected by the language compiler they are not as serious a problem and are more easily corrected. Semantic bugs are more

difficult to detect in the output, more difficult to pinpoint within the program, and often more difficult to correct.

In their description of debugging, Cassel and Swanson (1980) discuss syntax and semantic errors, and add a third category, logic errors. A program may be free of syntax and semantic errors but still not do what is intended because of incorrect logic. Cassel and Swanson suggest that the first and easiest step in debugging is desk checking. Programs should be planned and written on paper and should be carefully checked for syntax errors, duplication or omission, and sequence, prior to entering into the computer. Cassel and Swanson group syntax and semantic errors together, indicating that most of these errors are identified by the interpreter. They describe the use of test data to check every statement of code in the program. Tracing the program logic is discussed as either a manual or an automatic procedure.

Gates (1983) refers to a structured approach to BASIC debugging. He draws attention first to the resources available from the system such as BREAK, PRINT, STOP, CONTINUE, TRACE and FREEZE. Many of these are also available in Logo. Gates also stresses the importance of thoroughly defining the problem by knowing what the program does, when it does it, and what it should have done. He considers an exact problem definition as prerequisite to any debugging.

Bruce (1980) does not specifically discuss or define types of bugs, but deals with many debugging techniques and methods. In the introduction to his book, Bruce mentions one debugging technique so powerful, yet so simple that he feels it would offend readers to see a

chapter allocated to it. He feels that by explaining a program to someone else, the programmer will himself frequently catch the syntax and semantic bugs. His book, written about debugging in BASIC, presents many good techniques and strategies for handling logic errors. Some of these techniques may be applicable to the Logo situation as well. Bruce discusses both flowcharting and comment statements as preventative debugging. Flowcharting helps to provide for a clear and organized plan for proper program construction. Informative remarks used to label code are very important. Playing computer is described as the most accurate method of debugging. This requires going through the code step by step, doing with a pencil and paper what the computer does electronically. Block debugging refers to the examination of each subroutine apart from the main program. Structured or modular programming lends itself to block debugging. Designing-in (joining blocks or modules together) relates to the problems which sometimes occur when tested blocks do not function properly as part of the whole and is another source of bugs. Bruce (1980) concludes his book by saying;

> The many elements of the programmer's craft must work together in a successful debugging effort. No single technique is perfect for all applications and no method has any greater claim to fame than any other method. But used in concert with one another, and under the guidance of a logical plan of action, a few simple techniques are all that are required to debug even the most complicated of programs. (p. 339)

Types of Bugs and Debugging Strategies in Logo

The process of debugging is accepted by computer programmers as a normal part of programming. Is it accepted by teachers and elementary school students? Papert (1980) suggests that children are inhibited in their learning by the "got it" or "got it wrong" model, and that good learning strategies can be encouraged by following the debugging strategies of computer programmers. Elementary school teachers are not striving to make master programmers of their students. As computer software improves and computer applications increase, programming is receiving less emphasis. In spite of this, educators who have worked with Logo see merit in spending time on programming with Logo. Groen (in Sorkin, 1984) says, "what is learned in Logo is not primarily a programming language . . . but a way of establishing correspondences between a concrete world and one of abstract representations" (p. 50-51).

Some of the early sources of information on Logo result from the Brookline Logo Project which was conducted in an elementary school in Brookline, Massachusetts during the 1977/78 school year. In the Final Report of the Brookline Logo Project Part III: Profiles of Individual Student's Works, Daniel Watt (1979) reports the work of 16 grade six students who had between 20 and 40 hours of hands-on Logo experience. The discussion and teacher profiles on each of the students contain many references to bugs and debugging. The report is written in 16 parts with no attempt to synthesize or draw conclusions. Bugs and debugging strategies are not classified or looked at as a topic for

discussion. Sufficient information is included to cause one to want a framework to use for further analysis of this topic.

Kieren (1984) discusses three levels at which debugging in Logo occurs. He refers to these as the visual screen level, the list editing level, and the logic level. The first level, which is seen as an undoing action for the undesirable action or bug is used by young students working in immediate mode and likely by all Logo users in the beginning stages of Logo programming. The list editing level follows once students begin writing procedures and are able to relate a segment of the procedure to the resulting action. The logic level is the most sophisticated, most analytic, and occurs with students at a more advanced stage of programming.

Hillel (1985) conducted a study with 16 children, aged 7 - 10, who had no previous exposure to Logo. He found that children operated mostly in immediate mode and adjusted for errors, or according to Kieren's levels of debugging, were at level one. Hillel enumerates a number of difficulties or errors children experienced. (Hillel chooses not to use the terms bugs and debugging since most work was done in immediate mode and not in a program.) He grouped "errors" as those related to command meaning, those related to command inputs, those related to interface, those related to screen fit, and those related to sequencing of commands. In the final phase of his study Hillel indicates that 65 of 134 episodes were executed in procedural mode. Very seldom (only 18 instances) did debugging occur in procedural mode.

Leron (1985) reports on four years experience with grade six children using Logo in Israel. He stresses the role of the teacher in the Logo program and says that by using a quasi-Piagetian learning approach "the teacher can now concentrate on helping students reflect on and verbalize (and thus, conceptualize) their computer experiences" (p. 28). Students resist the suggestion of planning ahead or looking back and structured programming. He felt that the need for structured programming was not clearly evident to the students due to the lack of complexity of the projects. Debugging is of utmost importance and a better indicator of understanding than the actual writing of the program. He discusses the problems of long, unstructured procedures, interfacing procedures, and playing turtle.

Solomon (1975) suggests the mirror image problem as a cause of bugs. Not understanding the idea of state transparent (starting and stopping in the same position with the same heading) is another cause of bugs (Solomon 1976, Watt, 1979). The start up state (Solomon, 1976), interfacing two procedures (Gorman, 1983), and the positioning of the conditional command (Watt, 1983) are also cited as areas from which bugs arise.

Solomon (1975) provides some interesting ideas on debugging. She stresses learning to recognize and appreciate bugs, and suggests the idea of a bug collection as a valuable aid for a child to develop. The bug collection is a trouble shooting check list that is collected by each child and not something found in a book. Playing turtle is also suggested by Solomon as a powerful debugging strategy. This is confirmed by a number of other Logo writers, including Watt (1983).

Solomon comments on the value of student made aids for debugging. Many authors of books on Logo are including suggestions for manipulative aids for student use.

Gorman (1983) lists some bugs and gives some suggestions for debugging. He discusses the reading, understanding, and use of the Logo language messages, and the use of PAUSE, TRACEBACK, and WAIT. Nelson (1985) describes the merits of using STEP and TRACE as dynamic debugging tools, and especially PAUSE which is a powerful, interactive tool. Weintraub (1985) is of the opinion that high school students do not make adequate use of error messages.

Sugarman (1982) reinforces for Logo the idea put forth by Bruce (1980) for BASIC, that there is value in explaining the program to another, or even "thinking aloud". This task of defining the problem or explaining the bug helps the programmer to become a more involved and reflective participant in the debugging process. The solution often comes without any outside assistance.

Attitude of Students in Relation to Debugging

"Errors benefit us because they lead us to study what happened, to understand what went wrong, and, through understanding, to fix it" (Papert, 1980, p. 114). "It is this rethinking - the debugging process - that helps children to become clear and precise thinkers" (Martin, Bearden, & Muller, 1982, p. 58). The shift in emphasis from the final product or answer to the process of obtaining that product is evident and considered very important. People have considered children's growing tolerance for errors to be a change in attitude. Papert (1980) suggests that in "the LOGO environment, children learn

that the teacher too is a learner, and that everyone learns from mistakes" (p. 114).

Perlman (1976) points out that children should be allowed to work at the computer at their own pace and develop the attitude that an error is not a mistake but rather a challenge or a problem to be solved. Debugging is a valuable skill and programming the computer to solve a problem requires a deeper understanding than simply solving the problem by oneself.

Breen (1984) also endorses these ideas when he says, "In the Logo environment the child is not criticized for an error in drawing. The question to ask about a program is not whether it is right or wrong, but if it is fixable. The process of debugging is a normal part of the process of understanding a program. The programmer is encouraged to study the bug rather than forget the error" (p. 118).

Solomon (1975) supports the idea that bugs and debugging "are a part of life, not occasional accidents, not a sort of plague. We live with them and learn to like living with them" (p. 4). Solomon sees value in bugs because students can learn from them, and they often add interest.

The development of a more positive attitude toward errors is listed as one of the powerful ideas of Logo by a number of authors. Krasnor & Mitterer (1984) write, "Programming bugs are recognized as being unavoidable and useful in providing information. Such a reconceptualization of failure implies that the child's sense of mastery and motivation would not be negatively affected by errors" (p. 134). A sense of mastery and competence may be developed by children

since most bugs in Logo stand out, are debugged right away by the student, and feedback is provided. The child develops a feeling of power and success.

Knowing whether a program is worthy of debugging is a decision that students should make, sometimes with advice from the teacher. Solomon (1976) addresses the idea of teacher intervention in the debugging situation. She follows the "principle of allowing children enough success soon enough to make the fight worth while" (p. 1052).

Pea (in Sorkin, 1984) suggests that the guidance and support of the teacher are necessary if Logo is to be used as the designers hoped. He states further:

> A functional approach to programming recognizes that we need to create a culture for Logo in which students, peers and teachers talk about thinking skills, display them aloud for others to share and learn from, a culture that continually reveals how programming is a vehicle for learning general thinking skills, and that builds bridges to thinking about other domains of school and life. (p. 59-60)

Barton (1984) supports the importance of pupils being involved in debugging and accepting errors as correctable, necessary, and an integral part of the problem solving - learning process. She also stresses the need for person to person interaction and the teacher being involved with the student in the debugging process. "The teacher can focus on understanding the child's errors, on probing, questioning, clarifying and guiding thinking" (p. 18). "The child becomes aware of and focuses on his own thinking processes and

strategies and evaluates ideas" (p. 18). Barton reminds us that "attitudes, strategies, thinking and problem solving" (p. 18) develop slowly and with experience.

Summary

The area of bugs and debugging has been researched in considerable detail in computer languages such as BASIC. Bugs have been identified according to type and detailed information on strategies for debugging is available. The same is not true for Logo. Although bugs and debugging receive mention in many Logo articles, and authors stress the importance of debugging, the classification of bugs and a comprehensive description of strategies does not appear to be available. A considerable portion of the Logo programming time involves debugging. What are the debugging strategies that are used by elementary school students? Are students deliberately encouraged to improve their debugging strategies? Reference is made to the importance of student attitude regarding debugging by Papert, Solomon, and others. This suggests, therefore, that there is a need to acquire information in the area of bugs, debugging strategies, and attitude in relation to Logo.

Definitions

The following are definitions of terms as they are used in this study:

bug - an error which prevents a program from running or causes a program to function incorrectly.

debugging - the process of locating and correcting program errors; editing to allow for additions, deletions, or corrections.

Logo - a computer language which provides graphics, list and word processing, and other features. Predefined primitives are included and the user is able to "teach" the computer other terms.

Logo turtle - a little triangle that appears at the center of the screen (HOME) and can be made to draw by giving commands of distance and direction.

turtle graphics - those drawings done by the Logo turtle.

primitives - Logo's commands or basic instructions.

procedures - the new commands taught to the turtle so it will perform the tasks desired.

procedural mode - groups of commands are entered into the computer's memory and are stored there ready to be executed when and as often as requested.

immediate mode - each command or line of commands is entered and then executed as soon as the RETURN key is pressed.

structured programming - a way of ordering a computer program so that it is easily understood by dividing a long program into modules or small parts.

syntax - spelling of words and abbreviations, punctuation of the statements in the language including spacing and brackets, and the correct order or form.

semantics - understanding the meaning of the words and symbols.

logic - the patterns for doing things in a specific order, the series of organized steps to solving a problem.

attitude - how students think and feel about debugging computer programs, measured as positive, neutral, or negative in the areas of evaluation, usefulness, and difficulty, according to scales adapted from Nyberg and Clark (1979).

Limitations

The following factors limit the interpretation and the generalizability of the findings from the data:

1. The effect of the presence of the researcher, the taping, questioning, and note-taking.
2. The bias of the researcher in the observation and recording of student activity and behavior.
3. The size and nature of the sample, including the prior knowledge and attitude of the students.
4. The variation or lack of variation in the assigned tasks, which may not be representative of the generally assigned Logo tasks.

Delimitations

The exploratory study was restricted to:

1. A sample of 20 grade 6 students in 3 schools.
2. Logo turtle graphics.

Significance

The increasing impact of computer technology on the lives of elementary school students necessitates a careful examination of the

strategies used and the attitudes held by students as they work with computers. Logo is a computer language that is rapidly gaining acceptance in elementary schools. Papert (1980) describes Logo as an "object-to-think-with" (p. 23). If Logo programs in the schools are going to effectively contribute towards the development of thinking skills of students, the difficulties students face with Logo and the strategies for overcoming these difficulties must be understood.

A knowledge of the common bugs students encounter will help teachers plan experiences that will capitalize on bugs, and use them as a basis for teaching related concepts. Furthermore, teachers may want to explicitly teach strategies for overcoming bugs. A knowledge of the kind of strategies students spontaneously use, as well as those less commonly but effectively used, may help teachers plan the teaching of debugging strategies.

Outline of the Study

The purpose and background to the study were explained in this chapter. Definition, limitations, delimitations and significance of this descriptive study were also included.

Chapter II contains a description of the design of the investigation. The findings of the study are reported in detail in Chapter III. Chapter IV includes a summary of the investigation, a discussion of the findings and their implications, a list of conclusions, and recommendations for both research and teaching.

CHAPTER II

RESEARCH DESIGN AND SETTING

In this chapter the design of the investigation is presented. A detailed description of the three school settings is followed by information on the students in the study. The Logo programming assignments provided by the classroom teacher are then discussed.

Design of the Investigation

This descriptive study was designed for discovery. The researcher presented herself as someone wanting to find out from the students the understandings, thoughts and feelings they hold as they program in Logo, by observing their experiences and recording their discussions.

A number of measures were taken to enhance credibility and control for the influence of bias and error. A pilot study was conducted to better prepare the researcher for the observation sessions. The number of students observed and the amount of time spent with each student in the study provided an opportunity for rapport to develop and for prolonged data gathering to take place. A number of information sources, including observation notes, tape recordings, a written questionnaire, interviews with students and teachers, and disk files were used to provide for triangulation of data. Log notes were kept by the researcher, as well as samples of student's notes and copies of the teacher's written assignments. These will be referred to in subsequent sections.

Sample

The sample for this study was a group of grade 6 students, drawn from three elementary schools within the Edmonton Public Schools (EPS). Schools were chosen with the assistance of the EPS computer consultants. Criteria for selection were the involvement of the students in an ongoing Logo program and the willingness of the teacher to participate and have his/her students involved in observation and interviews. A description of the three school settings and the students involved in the study will follow in a later section.

It was expected that grade 6 students would be at different levels in Logo programming, and would produce work that would display computer bugs. It was also assumed that different types of bugs and different strategies for debugging would be evident.

Informal Pilot Study of Procedures

An informal pilot study was conducted. During the pilot study the work of students was observed and a data collection form designed and tested as an observation tool. This form, entitled Observation Sheet, appears in Appendix 1. Experience was gained in observing and recording the activities of the students and then using the observation notes to identify bugs and debugging strategies.

Observation Sheet

The Observation Sheet, containing four parts, provided space for recording identification data, the actual observation notes, the classification of bugs found in the observation, and the strategies identified. Identification data included student name, date, time, setting, and project. A six point check list was also included, but

only three of these were used consistently. The section for actual observation notes provided room to record the keyboard input of the students, the testing and debugging of procedures written, the comments, and actions of the students. These were recorded as fully as possible, noting whether the student was working in immediate mode or using the editor, and recording the testing of procedures written. Comments of the students were noted as well as related activities such as hand and body movements pertaining to the turtle graphics on the screen and use of pencil and paper or other materials. The sections of the form for the classification of bugs and debugging strategies were completed following the observation time.

The Attitude Scale

The "School Subjects Attitude Scales", developed by Nyberg and Clarke (1979) were adapted by the researcher and used to gain insight into the attitudes held by students regarding the debugging of their computer programs. The adapted scale, entitled Debugging Computer Programs, is found in Appendix 2. It consists of 24 bipolar adjective pairs which form the total scale. These are grouped into three sub-scales of eight pairs each. The sub-scales provide a measure of evaluation, usefulness, and difficulty. On the original scales, test-retest reliabilities with a one week interval, done on students in various subjects and from grades 5 through 11, fell in the 0.70 to 0.80 range. The reliability for internal consistency on the evaluation , usefulness, and difficulty sub-scales were reported to be 0.91, 0.90, and 0.82 respectively.

In addition to the Nyberg and Clarke scales, students were interviewed to obtain further information on attitude, as well as opinions on the bugs and debugging strategies students say they have and use. The questions used in this interview are found in Appendix 3. Notes on attitude were also kept in the log and on the observation sheets.

Procedure

Once teachers at the grade 6 level were recommended by the EPS computer consultants, contact was made by the researcher. The three teachers contacted expressed a willingness to participate in the study and did in fact have students actively involved in Logo. The fact that two of these classrooms were split grade 5 and 6 rooms was not seen as a problem, since each room had sufficient grade 6 students to meet the desired minimum of five students per room. The teachers, one female and two male, are all experienced teachers, have been involved with Logo for at least two years, and are currently members of a university curriculum and instruction course on Logo and mathematics. Each of the three schools was visited, making contact with the principal and the teacher involved. Discussion was held regarding the intent of the reseacher and the study, scheduling of visits, and procedures and policies to be followed. The researcher provided the principals with a letter of explanation and an approval form (Appendix 4) for the parents of students in the study.

Phase One. The initial visit to each class was an informal time for the purpose of becoming acquainted with the setting, the students, and the types of programming activities being conducted.

A minimum of 15 students, 5 from each school, were to be selected for detailed observation. Since the class in School A had only 7 grade 6 students all 7 were observed, and 7 students were also observed in each of the other schools. Random selection was not used in the other schools, but rather teachers were requested to choose students they felt would give the researcher a variety of programming approaches. All students were actively involved in Logo procedure writing and able to use the Logo editor. Specific attention was not paid to the type of student but to the programming work produced, since the purpose of this study was to identify those bugs that occur in students' work and the strategies students use in debugging. The type of student did in fact vary greatly as will be noted in the section on students which follows later in this chapter.

Following identification of the students, the researcher met with the students as a group for approximately 30 minutes. During this time the researcher introduced herself and the study to the students. The study was explained very briefly as an observation of the Logo activities of the students, without mention of bugs, debugging, or attitude. Students introduced themselves and were given opportunity to comment or ask questions. Students readily agreed to the researcher taking notes, using the tape recorder, and saving student programs on computer disk. All students appeared pleased to be involved, and anxious to demonstrate their abilities in Logo. Students completed an information form, giving their name and age, school and outside interests, and their thoughts on computers and Logo.

Phase Two. Students were observed individually during their hands-on computer time in an attempt to capture as many examples of bugs and debugging as possible. During this observation the data collection form was used to record the activity of the students so bugs and the debugging strategies used could be identified later. A tape recorder was also used when possible and students encouraged to think aloud as they debugged in order to more clearly reveal their debugging strategy. Taping was not always appropriate due to the variety of other activities taking place in the classroom.

Procedures written were saved on disk by the researcher with the permission of the student. These were saved both during and after debugging and used to increase the reliability of the data collection forms. Disk files were also examined for evidence of debugging that showed attempts to expand, refine, explore, make more efficient, make more universal, or make more attractive a particular procedure.

Log notes were maintained on students, particularly in regard to attitudes about bugs and debugging, expanding and refining, and programming style.

Amounts of time spent observing each student were not equal. Both the number of observation periods and the length of each period varied. Breaks were made to suit the ongoing classroom schedule, the students needs, and at appropriate points in the Logo program being written. The researcher continued observations until it was felt that data collected would provide sufficient samples of bugs and debugging strategies and a feel for the style of the programmer.

Phase Three. During the final visit to each school, attitude questionnaires were administered. Questionnaires were administered to all students in the three selected classes that were involved in Phase One and Phase Two. These questionnaires were designed by the researcher, based on the "School Subjects Attitude Scales" by Nyberg and Clarke (1979).

Following the administration of the attitude questionnaire to the class, the project students were interviewed individually. They were shown the printout of their Logo procedures, both listing and graphics. In some instances students were asked to further explain what they were doing, to provide the researcher with further insights into strategies used. Students were then asked to respond orally to a set of questions regarding bugs, strategies and attitudes. These interviews were tape recorded and later transcribed.

It was not until Phase Three that the focus on bugs and debugging strategies was made evident to the students. This was intentionally withheld so as not to influence the students.

Time Frame

All interviews and observations were conducted during the period from January 21 to February 26, 1985. The exact dates and length of each observation session are found in Appendix 5. Introductory sessions were held on January 21, January 24, and February 11 for Schools A, B, and C respectively, while concluding interviews were held on February 11, February 13 and February 26, 1985. An overview of the observation schedule is presented in Table 1. All names in

this table and throughout the report have been changed to assure anonymity of the subjects.

Table 1

Observation Schedule

School	Student	Obs 1	Obs 2	Obs 3	Obs 4	Obs 5	Obs 6	Total
A	Barry	30	25	30	40	50	25	3 h 20 m
A	Carol	30	40	20	40			2 h 10 m
A	Cindy	40	30	50	50			2 h 50 m
A	Connie	30	25	15				1 h 10 m
A	David	30	25	35				1 h 30 m
A	Don	25	30	10	20	20		1 h 45 m
A	Doug	25	25	25				1 h 15 m
B	Mark	20	30	25	30	35		2 h 20 m
B	Mike	40	35	25	20	35		2 h 35 m
B	Murray	15	25	25	30			1 h 35 m
B	Lana	20	30	30	25			1 h 45 m
B	Laura	30	20	40				1 h 30 m
B	Linda	45	35	55				2 h 15 m
B	*Lucy	10	20	15				
C	Ray	15	20	20	35	45		2 h 15 m
C	Sandy	25	35	30	40			2 h 10 m
C	Shauna	50	25	20	50			2 h 25 m
C	Ted	30	20	30	30			1 h 50 m
C	Trevor	25	30	50	35			2 h 25 m
C	Troy	40	20	35	15			1 h 50 m
C	Vicki	25	25	30	30			1 h 50 m

School	# Students	Total Time	Ave Time	Total Obs	Ave Obs
A	7	14 h	2 h	28	4/student
B	6	12 h	2 h	24	4/student
C	7	14 h 40 m	2 h 5 m	29	4/student
A+B+C	20	40 h 40 m	2 h 2 m	81	4/student

* Lucy was observed three times, but due to difficulties discussed in the text, information on her observations is not included in any of the statistics or analysis.

Adjustments to the Design

Deletion of One Set of Observations. The seventh student in School B worked only in immediate mode throughout the first session and came with notes she had obtained from another student for the next session. When the notes were entered and did not give the desired results she became very flustered. She tried to construct a procedure herself but seemed very uncomfortable and was therefore excused. Following the second observation period and consultation with the teacher, it was decided not to include this student in the study as she was obviously nervous and having considerable difficulty carrying out the assignments given by her teacher. The student was not aware that her work was not being included and she was seen for a third session by the researcher at her request. Since there was no meaningful data on her, she is not referred to again in this study.

Interventions By the Researcher. The "thinking aloud" technique was used to gain information on the strategies students used. When encouraged to think aloud some students began to question the researcher. An attempt was made to keep interventions to a minimum. On a few occasions it was felt necessary to assist, as the classroom teacher was not available, and the student was either insistent or ready to give up or delete the program. At the end of one session, Carol asked for help in making triangles and circles and it was provided. On one occasion the researcher suggested to Cindy that she do what she was asking the turtle to do (play turtle). Mike was given assistance with his recursive procedure when he wished to have the design repeat five times.

The researcher did provide assistance on a number of occasions in regard to disk and file handling and on using the editor. It was felt that these instances would not adversely affect the data for the research.

Settings

The 3 classrooms that were chosen for this study provided fairly diverse settings. Each of the schools was located in a different sector of the city. There was also variation in the size and age of the school. The 3 principals and the 3 teachers willingly participated in the study and were most cooperative with the researcher.

School A

The classroom in School A was a split grade 5 and 6 classroom. Since there were only 7 grade 6 students in the room, it was decided that all 7 students be included in the study. For most of the duration of the study this classroom had two Apple computers located in the room. A printer was attached to one of the computers. Near the end of the study only one computer remained in the room as the other was required in another classroom in this fairly large school. The computers were located on opposite sides of the room, on tables up against the wall. The computer which was used throughout the study was obscured from the view of most of the students by a portable room divider. Near one computer was a list which indicated the rotation system for having turns to work at the computer. The Terrapin version of Logo was being used.

The students were accustomed to taking turns, working throughout most of the day, with one student at a time at each machine. Logo assignments were introduced briefly following the opening exercises on two of the four mornings the researcher was present. Students appeared to work very independently at the computer with little interaction between themselves and other classmates or the teacher.

School B

In School B the grade 6 students also shared a classroom with grade 5 students. There were 14 grade 6 students in the class and 6 of them participated in the study. The observations took place in a relatively unused room adjoining the regular classroom. Students came on a rotational basis to the computer in this adjoining room. Four, sometimes five, Apple computers, using the Terrapin version of Logo, were located around the periphery of the regular classroom. Students made use of the computers during part of the school day, and occasionally during lunch hour in a club setting. The lunch hour sessions provided pupils an opportunity to share ideas, show others their programs, consult with their teacher, or play computer games. Some of the students received instruction and assistance with Logo in a small group setting during school time on a rather irregular basis. The classroom walls contained a number of displays relating to computers and to Logo. Logo primitives with illustrations were displayed.

School C

Seven members from this large class of 32 grade 6 students were observed by the researcher. This classroom was a semi-open area with

walls on two and one-half sides. The back of the classroom was open to the library area and the partial wall divided the classroom from the adjoining computer laboratory. Groups of students from this classroom and other classrooms made use of the computer lab on a regular basis. Throughout the study students were observed by the researcher at a computer set off to one end in the lab. During observations the lab was usually being used by groups of 10 to 20 students. The lab contained five Apple computers, five Atari computers, and a printer. Many of the students were accustomed to using either Atari or Apple Logo (LCSI) or Apple Sprite Logo. Students used Apple Logo during observation times.

The students observed in this school were receiving some instruction, assignments, hands-on computer time, and opportunity for computer club time, on a fairly regular basis. This varied from student to student, with a group of students receiving some computer time while others were receiving instruction in French. Due to the convenient availability of the computer lab, students from this class would sometimes be permitted to make use of the computers when they were not in use by others in the school and when students had satisfactorily completed their other classroom work. Students could obtain a pass when they wished to be part of the computer club session. The room was available for those with club passes each morning and noon hour. Both students and teachers used this time for Logo and also word processing activities. Occasionally a teacher was available to provide assistance, but generally this was a fairly relaxed time for exploration and sharing. Some students used the time

to complete assignments expected by their homeroom teacher. The actual amount of computer time received by each student in the observation group in this school was not easy to determine.

The computers in the lab were arranged around the perimeter of the room, with most on tables and others in carrols. Two of the machines had large monitors which were sometimes used for class demonstrations. This room was decorated with computer pictures and posters, charts with Logo primitives and information, and samples of student work.

The Students in the Study

A total of 20 students participated in the study, 7 students from each of 2 elementary schools, and 6 from a third elementary school, all from within the Edmonton Public School system. The researcher requested that the teachers choose students who were writing procedures and able to use the Logo editor to the degree that they could edit the procedures they were writing.

Although the focus of this study is not on the student, but on the bugs and debugging strategies present in the Logo procedures written by the students, the following information is provided as evidence that the sample included a variety of student abilities, interests, and backgrounds.

The 7 students from School A comprised the total group of grade 6 students in their split 5/6 classroom. The teacher classed these students as generally average ability students. The available

intelligence and mathematics scores are listed in Table 2. Recent standardized reading scores were not available.

Of the 14 grade 6 students in School B, 6 students participated in the study. These students represented a wide range of abilities as indicated by their teacher and evidenced in the scores recorded in Table 2.

The 7 students chosen from a class of 32 students in School C included some of the best Logo programmers as well as two of the weakest Logo programmers in the class. Interestingly enough, Ray, the

Table 2

IQ and Mathematics Scores

School	Name	IQ-V	IQ-NV	IQ-Q	EPS Math
A	Barry	113	98	89	NA
A	Carol	126	96	113	49
A	Cindy	94	87	93	19
A	Connie	111*	113*		NA
A	David	99	87	90	10
A	Don	115	115	108	49
A	Doug	86	87	110	33
B	Mark	127	129	115	66
B	Mike	91	89	94	73
B	Murray	117	111	132	84
B	Lana	90	89	94	13
B	Laura	111	98	96	4
B	Linda	105	100	94	10
C	Ray	109	108	106	70
C	Sandy	98	102	108	NA
C	Shauna	76	84	82	40
C	Ted	124	136	118	89
C	Trevor	117	130	141	97
C	Troy	104	111	128	56
C	Vicki	117	91	112	63

IQ Canadian Cognitive Ability Test (Verbal, Nonverbal, and Quantitative)
* Lorge Thorndike Test
** End of grade 5 mastery test

student felt by the teacher to be the strongest and most productive Logo programmer has fairly average scores on the tests included in Table 2, and was, until recently, considered to be a difficult student and a discipline problem. This was his first year in a computer program.

The group of 20 students selected contained 9 girls and 11 boys. (The original group had 10 girls and 11 boys.) Sixteen of the students were 11-year-olds, while 4 (all girls) were 12-year-olds. Nine of the students were in their first year of Logo work. Although the other 11 had had exposure to Logo prior to their grade 6 year, none of them had an extensive Logo background. For many of these students their experience was very limited due to the limited access to computer hardware.

Programming Assignments

The programming assignments required of the students varied in each of the 3 schools. These assignments will be discussed in terms of the description given by the teacher. They will then be discussed in terms of their interpretation by the students.

Assignments Provided By the Teacher

Students in School A had been given an assignment prior to the arrival of the researcher. Some students had started on the assignment, while others had not begun actual work on it. The students had been asked to write a procedure which made use of three subprocedures. The contents of the procedures were left to the discretion of the students. Upon completion of this assignment students were to write a procedure or procedures to illustrate

symmetry. The teacher spent approximately 5 minutes discussing the idea of symmetry. The line or axis of symmetry was identified. Students were asked to draw a shape, make changes to the left and right commands, and obtain a symmetrical copy. The following example was given on the board:

```
REPEAT 4 [FD 30 RT 90]
REPEAT 4 [FD 30 LT 90]
```

Geometric shapes as well as symmetry in nature were discussed. Students were asked to give examples in nature, and responded with some non-nature examples as well as nature examples such as leaf, bird, and butterfly.

Students in School B were observed during the latter part of January and the first half of February. They had been asked to create Valentine's Day designs.

Students in School C were provided with written assignments. Copies of these assignments are provided in Appendix 6. Upon the arrival of the researcher, a group of students was observed working with an assignment called Exploring Powerful Ideas, which contained written information and two diagrams. The assignment dealt with recursion, variables and conditional statements and gave four questions to direct the exploration. A second assignment, entitled Logo Challenge, had been written out by the students. The students were later requested by the teacher to work on it during their computer turn. The Logo Challenge assignment was intended by the teacher to be a lead-in to the idea of tessellations, and it structured a number of tasks. The tasks included writing procedures

to draw rectangles and squares using variables, using the procedures to construct a large rectangle and then fill it with small squares, using the procedures to design an even larger rectangle and filling it with even smaller squares, and writing procedures to position the turtle. The last task on positioning the turtle requested the students to try using the REPEAT command, variables, and descriptive variable names. The third assignment distributed to the students in School C during the time the researcher was there was entitled Exploring Tessellations, and contained a written description as well as a computer printed graphic. Students were challenged to construct procedures that would provide a design similar to the one pictured on the assignment sheet, and were reminded of the wisdom of breaking a large problem into smaller problems. They were further challenged to see what other shapes they could tessellate, and also to tessellate combinations of shapes.

The assignments provided in all three schools were very open ended and allowed for pupil exploration, discovery, and creativity. The assignments in Schools A and B were the least structured as to the use of the Logo language (procedures and subprocedures being the only ideas voiced), and also fairly unstructured as to theme (symmetry and Valentine's Day). The written assignments of School C were also very open ended, as demonstrated by the results of the students, but did provide definite content themes as well as Logo language ideas to try. Concepts from the Logo language included were procedure, recursion, variable, conditional statements, changing conditions, changing variables, incrementing variables, building blocks, inputs, distance,

angle, REPEAT, descriptive names, positioning the turtle, breaking down problems, using patterns, and extending the results to new but related situations. Themes included exploration with a provided FAN procedure, covering rectangles with squares (ideas of tessellation and area), and tessellations.

Assignments Interpreted By the Students

School A. In dealing with the assignment requiring three subprocedures in a superprocedure, 5 of the 7 students did not succeed with the task as set out by the teacher. Three of the boys wrote three procedures, but made no attempt to include them in a superprocedure. The fourth boy wrote one long procedure. One of the girls wrote two procedures. One of her procedures did in fact have three parts to it, three squares placed in different locations on the screen. Two of the girls did succeed in completing the task as assigned. One girl wrote the three subprocedures (a rectangle, a square and a T - named YOU, FOOTBALL, and TE) and at the close of that session stated that she was not sure how to put it all together. She returned the next day to "write a big procedure" and did it in a fashion not at all unlike what one would expect a student to do on a first attempt. Her continued efforts met with success. The other student, also a girl, wrote two identical (except for the names - JA and JA2) subprocedures, which were to be the eyes in the face generated by the superprocedure (MJ). During my last observation with this student, she was still working on this task, and at that time the superprocedure was calling seven subprocedures. It is important to note that the terms "not achieve" and "succeed" are only the

researcher's terms. As far as all of the students were concerned they had succeeded and proudly saved the results of their work on disk.

Six of the 7 students worked on the symmetry assignment. One student made two attempts, one of a butterfly and the other of a "square inside a square". Although he enjoyed his exploration, he was aware that he had not completed the task. He seemed not to have a clear understanding of the idea of symmetry or of the Logo primitives and procedures required to carry out his plan. One boy drew a fairly complex face design on paper, but met with construction difficulties. His task was not completed during the observation period. Two simple square subprocedures (like the illustration the teacher gave) were written by one of the girls. When called, one following the other, a symmetrical design resulted. Although this student was one who had previously used subprocedures and superprocedures, these were not joined together in a superprocedure. The other 3 students, a girl and 2 boys, each completed a procedure illustrating symmetry which made use of squares arranged in varying degrees of complexity. The 2 boys also worked on designs involving circles, in which they found it more difficult to maintain the idea of symmetry. These two tasks were not complete when the last observation was made.

School B. The task of making a Valentine's Day design resulted in some interesting displays. These varied greatly in complexity. One of the girls chose to design large letters spelling the word LOVE. Since she was not sure how to do diagonal lines she made the letter U instead of V. Upon completion of this project, she began designing a large card. Another girl first made a large rectangle, divided into

two large side by side squares, and then proceded to design large letters to spell out BE MY VALENTINE. When the last observation took place she was about to begin the first N. Both of these girls had a "spaghetti" style of programming where everything went into one long procedure, each command on a separate line. Although there was no sign of modular programming, they did actually work in modules, doing a short section (usually a letter at a time), testing and debugging that section before adding the next part. One of the boys designed the letters BE MINE. He put each word in a subprocedure and then wrote a superprocedure, BM, that called the two subprocedures, BE and MINE.

One of the girls was first observed finishing her Christmas design, in which she had drawn a cross, moved the turtle to the top left corner and designed the letter P. She wrote subprocedures for the letters E, A, and C, and then added them to the superprocedure called START. This superprocedure called six different subprocedures (E being called twice). Upon the completion of this task, the Valentine's project was undertaken. The word LOUE was written in large letters (U was substituted for V to simplify the task), starting in the upper left and proceding to the lower right corner of the screen. Each letter was contained in its own procedure, and then the superprocedure LOUE called the four procedures. All of the necessary moves between letters were included in the letter procedures.

The project completed by another student included the word LOVE across the top portion of the screen, accompanied by a large heart designed in a subprocedure, H, and called by the last line of the main

procedure. He then proceded to make a smaller heart and repeat it a number of times to form a pleasing design as his second project. This resulted in some interesting exploration with recursion. This student, not wishing to do another Valentine's project and unable to think of a project on his own, asked the researcher to suggest one. When the idea of symmetry, which had been used in the other school, was mentioned he went right to work on an interesting geometric display. He made squares to the left and right of HOME, inscribed irregular triangles, and then started to inscribe circles within the triangles.

The Valentine's interpretation of the other boy in School B was a large heart pierced through by Cupid's arrow with the word LOVE written under the heart. The superprocedure made the heart and called two subprodedures. One subprocedure (CUPID) made the barbed arrow, while the other subprocedure (LETTERS) set the turtle's position and called four subprocedures to design the letters L, O, V, and E. Upon completion of the project this student went on to design a St. Patrick's Day shamrock, and then started on an Easter egg.

School C. The students observed in the computer laboratory during the researcher's initial classroom visit to School C were working with the Exploring Powerful Ideas assignment sheet (Appendix 6). This was certainly an exploration type of situation, and several of the students obtained interesting and attractive screen displays as a result of their exploration. The explorations of one of the students was noted in more detail on two subsequent visits. Since his task was to explore and experiment, these sessions were not coded for

bugs and debugging strategies. Following these sessions, this student was observed working on the tessellations project. Because of the SCRUNCH setting the diagram on the assignment sheet did not appear to be a regular hexagon. Using the irregular hexagon he perceived made this task a difficult one. Also, his strategy of doing one horizontal and then one vertical line was an unexpected interpretation.

One student was observed for two short sessions as he worked on a project of his own choice. He designed a fairly elaborate castle. Following this he went onto the tessellation project. He saw the shape as a regular hexagon, built a row of hexagons across the bottom, and then stacked them, making one fewer in each row, until there was only one in the row. This student's next project was to tessellate triangles, this time attempting to fill the whole screen. Although this task was not complete when the last observation took place, a portion of it was done and many of the problems worked out.

One of the girls also designed a castle as a project of her choice. After spending two sessions on the castle, she too went on to the tessellation project, leaving the castle incomplete. The irregular hexagon procedure she wrote presented many problems when she attempted to tessellate the hexagon. A couple of very interesting designs emmerged but she was not successful in filling the screen with hexagons.

One student worked only on the Logo Challenge assignment (Appendix 6). He had considerable difficulty handling the variable inputs, but was successful and in the end completed both problems one and two on the assignment sheet.

Another student working on the same assignment met with difficulty in finding an origin point that was suitable so the whole rectangle would fit the screen. When she returned for her second session she chose to work on the tessellation assignment as she thought the first task was too difficult. Rather than tessellate hexagons, she chose to simplify the task and tessellate squares. She then went back to the Logo Challenge page and worked on problem three. This presented problems as she embedded the move procedure in the shape procedure, and then tried to use the shape procedure a second time. A subsequent session was spent on problem two from the same sheet. Although this task was not completed during the observation time, the student did succeed in locating the turtle so that the large rectangle would fit the screen.

Upon completing three projects of his choice, a sheet of graph paper, a tunnel of concentric squares, and a flower, one student then worked on tessellating irregular hexagons. After much difficulty and a couple of interesting designs, he finally was successful in getting a set of three hexagons joined to itself to form a set of six hexagons.

The other student began with the Logo Challenge introductory procedures and problem one. Following that she spent considerable time working with the tessellations assignment. She found this task very difficult but worked with persistence.

Adapting Assignments

The assignments as given by the teacher presented either a content or subject theme, or a Logo programming task, or both. All of

the assignments were open ended and allowed for considerable variety in interpretation by the students.

Students were able to adjust the assignment to their level of programming expertise, although the assignments from School A and School C did expect standards that were difficult for some students to attain. Students, nevertheless, were able to do something with the tasks. The two examples of the work of students in School B are given in Figures 1 and 2. They show the contrast in the complexity of the end product from the same assignment. The tessellation of squares in contrast to the tessellation of hexagons or triangles in School C is another example of students adjusting the difficulty level of the assignment to their level of ability and experience. Figures 3 and 4 show the work of two students on the tessellation task. Several students in School A changed the assignment, writing three separate procedures instead of a superprocedure calling three subprocedures.

On three occasions students abandoned the task. Doug's attempt to construct a symmetrical butterfly was one example. The teacher had assigned the topic of symmetry and he had chosen the task of making a butterfly. Although he had abandoned this task after working on it for one period, he did attempt another project related to symmetry, which was not complete at the time of the last observation. Vicki also worked on an assignment for one period and then started a new task the next time, saying that the other one was too hard. During the last observation period with Barry, he also abandoned a task to do some experimenting, saying that he would come back to it later.

On a number of occasions students were temporarily distracted from the original task because of what one might call serendipidy; a bug in the program resulted in an interesting and attractive design that had not been planned or sought after. Mike never returned to his original task, but rather spent time exploring the new found design. Figure 5 shows some of these designs.

Figure 1

Valentine Project by Laura

```
TO LOVE
BG 5
PC 4
LT 90
PU
FD 90
RT 90
PD
FD 50
BK 50
PD
RT 90
FD 30
PU
FD 10
LT 90
FD 50
RT 90
PD
FD 20
FD 5
RT 90
FD 50
RT 90
FD 25
RT 90
FD 50
RT 90
PU
FD 35
RT 90
PD
FD 50
LT 90
FD 25
LT 90
FD 50
PU
RT 90
FD 10
RT 90
PD
FD 50
BK 25
LT 90
FD 20
LT 90
FD 25
LT 90
FD 20
LT 90
FD 50
LT 90
FD 20
END
```

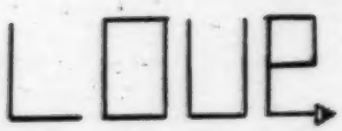


Figure 2

Valentine Design by Mike

```
TO L

HT  PC 0  LT 90  FD 125  RT 90  FD 110  PC 1  RT 180  FD 45  LT 90
FD 35  PC 0

FD 25  PC 1  LT 90  FD 45  RT 90  FD 35  RT 90  FD 45  RT 90  FD 35
RT 180  FD 35  PC 0

FD 25  LT 90  FD 45  PC 1  RT 140  FD 55  LT 103.5  FD 55  RT 140
PC 0  FD 50  LT 88  FD 25  LT 90

PC 1  FD 50  RT 90  FD 35  RT 180  FD 35  LT 90  FD 25  LT 90  FD 25
RT 180  FD 25  LT 90  FD 25  LT 90  FD 35

H

END

TO H

HT  PC 0  HOME  PC 1  REPEAT 230 [RT 1 FD 1]  PC 0  HOME  PC 1
REPEAT 230 [LT 1 FD 1]  FD 118 LT 797.5  FD 123

END
```

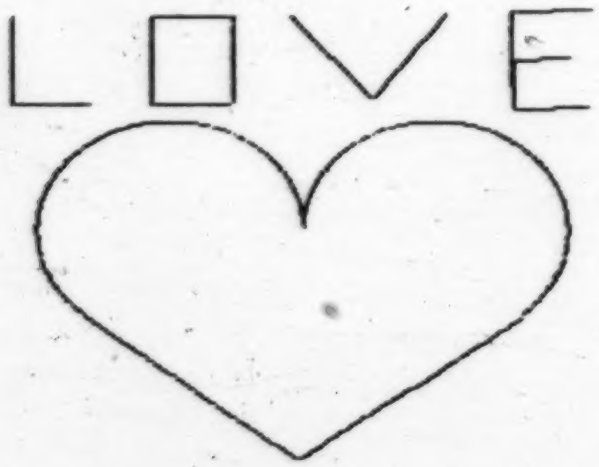


Figure 3

Tessellation of Squares by Vicki

```
TO SQARES
REPEAT 4 [FD 45 RT 90]
END

TO SQ
REPEAT 4 [RT 90 SQARES]
RT 90
FD 90
REPEAT 4 [RT 90 SQARES]
LT 90
FD 90
REPEAT 4 [RT 90 SQARES]
LT 90
FD 90
REPEAT 4 [RT 90 SQARES]
FD 90
REPEAT 4 [RT 90 SQARES]
LT 90
FD 90
REPEAT 4 [RT 90 SQARES]
END
```

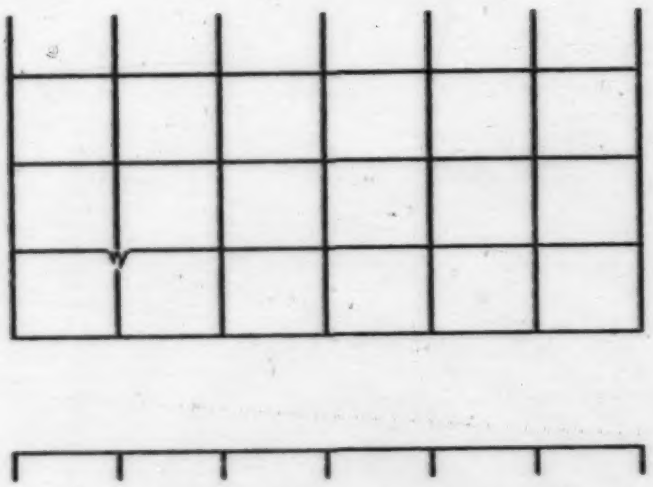


Figure 4

Tessellation of Hexagons by Ray

```
TO ALL
MOVE
REPEAT 5 [S MOVE2]
PU SETPOS [-95 -55] PD
REPEAT 4 [S MOVE2] FD 30
LT 90  PU  FD 175  PD  RT 90
PU FD 15 PD
REPEAT 3 [S MOVE2]
PU FD 30 LT 90 FD 122 RT 90 PU FD 15 PD
REPEAT 2 [S MOVE2]
PU FD 30 LT 90 FD 75 RT 90 PU FD 15 PD
S
END

TO MOVE
PU SETPOS [-120 -100] PD
END

TO S
REPEAT 6 [FD 30 RT 60]
END

TO MOVE2
PU RT 90 FD 50 LT 90 PD
END
```

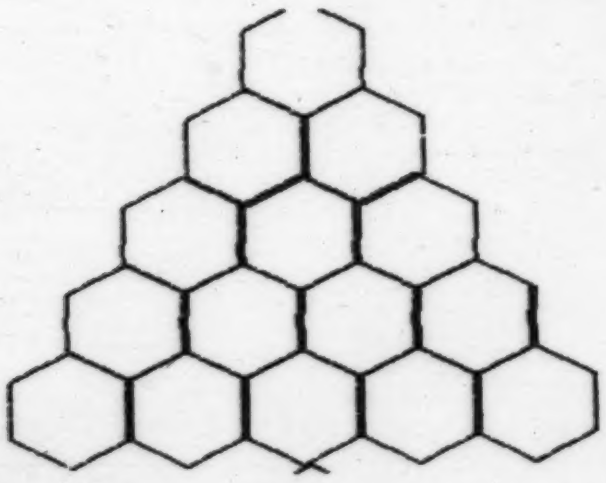
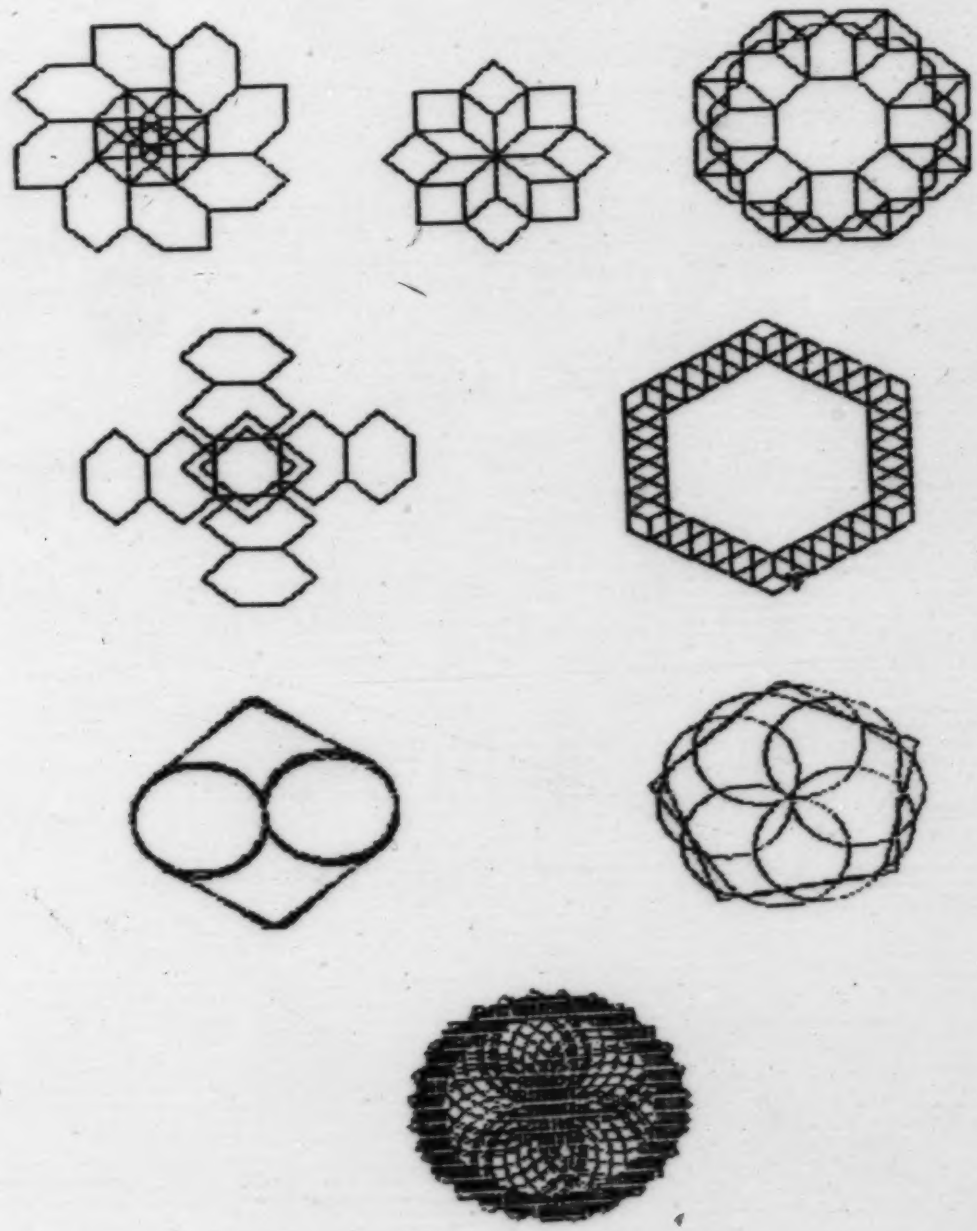


Figure 5

Serendipidy Designs from Bugs



Summary

This chapter described the design of the study, including the sample, pilot study, procedures followed, and instruments used. The setting in each school was described and information was given related to the students and the tasks they were given. Students' approaches to these tasks were summarized.

CHAPTER III

RESULTS OF THE INVESTIGATION

This chapter provides a description of the procedures used in analyzing the data and establishing the categories of bugs and debugging strategies. The categories of bugs are then described and information regarding their frequency is provided. This is followed by information on the debugging strategies and their frequency. A discussion then relates strategies to bugs. The students' attitudes toward bugs and debugging, as determined by the questionnaire, interview, and log notes are presented. A number of other related considerations are then discussed.

Data Analysis Procedures

At the completion of each day's observation, a hard copy was made of all the computer disk files that had been saved during and following the work of the students. The corresponding hard copy was then attached to the observation sheets for each student. Log notes written on each student observed and general school log notes were also typed following each observation session. Samples may be found in Appendix 7.

Analyzing the Observation Sheets

Using the observation sheets, the bugs which occurred in the students' work were identified and noted in the left hand column of the observation sheet. The computer disk contents were also checked to clarify and verify the notes recorded by the researcher. Debugging

strategies used by students were noted in the right hand column of the observation sheet.

Upon completion of all of the observations, the observation sheets were analyzed and all bugs and debugging strategies identified were tallied. The bugs were originally tallied under 25 headings. These were then grouped under 8 major headings. Strategies were first tallied from the observation sheets under 36 headings and then grouped under 14 major headings.

When identifying the debugging strategies, it was noted that more than one strategy was often used to get rid of a particular bug. On many occasions multiple attempts were also required. The time required to eradicate a particular bug also varied greatly. Although precise timings were not kept on each bug, the variations were very obvious. While some bugs were corrected almost instantly, others required nearly the whole observation period to correct.

Reliability Check

Verification of the identification and classification of bugs and identification of debugging strategies was done by having an expert in the field of Logo examine samples of the data collected. Eleven of the 81 observation sheets were examined. In an attempt to make the 11 observation sheets representative of the total, 3 were from School A, 4 from School B, and 4 from School C. Three were first observations, 3 were second observations, 3 were third observations, 1 was a fourth observation, and 1 was a fifth observation. All 11 were coded on a different student, with 5 girls and 6 boys being included. The 11 observation sheets verified contained 20.6% of the identified bugs and

19.0% of the debugging strategies. In addition to the verification of the above mentioned sheets, a number of other sheets were checked in part, and consultation was obtained where the researcher had questions.

In checking the papers, the examiner found no disagreement with the researcher on any of the identifications and classifications of bugs and strategies. The examiner did find five bugs and four strategies that the researcher had failed to code. These omissions were then added. Prior to reinstatement of the omissions there was a 97.3% level of accuracy on the part of the researcher. The results of the reliability check are found in Table 3.

Table 3

Reliability Check

Sch	Name	Obs	Total Bugs	Total Strat	Error Bugs	Error Strat	Omit Bugs	Omit Strat
A	Carol	1	3	7	0	0	0	0
A	Cindy	3	8	19	0	0	1	1
A	Barry	3	11	23	0	0	0	1
B	Mike	1	14	25	0	0	1	0
B	Laura	2	2	11	0	0	0	0
B	Linda	3	19	31	0	0	1	0
B	Mark	4	14	27	0	0	0	0
C	Sandy	1	7	21	0	0	1	1
C	Trevor	2	9	17	0	0	0	0
C	Troy	2	7	18	0	0	0	0
C	Ray	5	15	30	0	0	1	1
	Total		112	219	0	0	5	4

Percentage of Error or Omission - 2.7

Attitude Questionnaires

Attitude questionnaires on debugging computer programs were scored, total scores and sub-scales tallied, and transfered to a stem

and leaf display. The results of these scales were related to specific students and compared to their comments on how they felt about bugs and debugging, gathered from the interview tapes.

Bugs Observed in Students' Work

During the 40 hours and 40 minutes spent observing 20 students a total of 528 bugs were observed in the work of the students. These bugs were studied to identify similarities and patterns and were then grouped. The following major headings and sub-headings resulted.

1. Dimension
   - Length - of line, side of shape, between parts
   - Number of REPEATS
   - Circles
2. Orientation - amount of turn
3. Lateralization
   - RIGHT for LEFT
   - LEFT for RIGHT
4. Locating Origin
   - Locating starting position (corner)
   - Joining shapes together
   - Positioning parts
5. Omissions
   - PENUP - PENDOWN
   - Lines from notes
   - Forward and turn commands

6. Excess

   Forward and turn commands

   PENUP - PENDOWN

   Brackets

   Variables, variable inputs

7. Syntax

   Spacing

   Spelling

   Other typing errors

   Form - SETPOS, SETX, SETY

8. Miscellaneous

   Color combinations

   Semantics - PU for FORWARD , PR for pen right,
   EDIT, TO, REPEAT

   HOME, SETPOS, SETX, SETY - can't relocate shape

   Incorrect mathematics calculation

   Logo System Bug

Each observation session for each student in each school was charted, recording the bugs under the 25 sub-headings. These were grouped into the 8 major categories of bugs. The total number of errors made per session, per student, per school, and per category were calculated. A summary chart of these findings appears in Table 4. Each of the major categories is discussed in the following paragraphs. Samples of student work illustrating the various bugs are discussed under the same major headings in Appendix 8.

Table 4

Bugs Identified

NAME	1	2	3	4	5	6	7	8	Total
Barry	20	8	13	3	6	1	2	6	59
Carol	9	2	8	5	2	2			28
Cindy	13	3	3	3	1			1	24
Connie	1	1	2		2			2	8
David	5	1	2	1	4		1	1	15
Don	5	4	2	4	3				18
Doug		1	1		1	1	1	5	10
Mark	16	10	1	4	6		2	2	41
Mike	13	10	11	1	2		3	2	42
Murray	2	3	1		6				12
Lana	9	1	9		5				24
Laura	4	1	1		3		1	1	11
Linda	7	2	19	1	5		1		35
Ray	22	3	3	5	6	1	2	3	45
Sandy	7	4	6	4	3		1	5	30
Shauna	5	8	2	6	4	2	3	3	33
Ted	4	1		4	2		2	1	14
Trevor	6	2	8	4	3			1	24
Troy	8			1	6	7	4	3	29
Vicki	9	2	4	2	4		2	3	26
Sch. A	53	20	31	16	19	4	4	15	162
Sch. B	51	27	42	6	27	0	7	5	165
Sch. C	61	20	23	26	28	10	14	19	201
Total	165	67	96	48	74	14	25	39	528
Percent	31.2	12.7	18.2	9.1	14.0	2.7	4.7	7.4	100.0

Key to Bugs Identified

1. Dimension
2. Orientation
3. Lateralization
4. Locating Origin
5. Omissions
6. Excess
7. Syntax
8. Miscellaneous

Dimension Bugs

The largest number of bugs in a single major category was classed as dimension errors. Here there were 165 bugs or 31.2% of the total bugs identified. Ten percent of these were miscalculations in the number of REPEATs required in designing a shape (square, rectangle, circle, arc, etc.) or in determining the dimensions for a circle. Handling of the circle presented more bugs for those students using Terrapin Logo where there is not a predefined circle command. Twenty-one percent of the bugs related to miscalculations in the length of a line, in the length of the side of a shape, or the length of the distance between parts.

Orientation Bugs

Orientation bugs are those which relate to the heading of the turtle or the amount of the turn (angle) and account for 12.7% of the total bugs identified. The students observed made extensive use of 90 degree turns, which were useful in many of the constructions being done. Turns other than 90 degrees presented many students with difficulty. The less regular the shape the more likely the incidence of orientation bugs. Some of those students in School B who were doing lettering found diagonals difficult and were faced with bugs or avoided the conventional form of the letter, making a U for V and a double U instead of W. Orientation bugs were frequent in the hexagon tessellation work of students in School C. Some of these resulted from the inaccuracy of the hexagon diagram on the assignment sheet.

Lateralization Bugs

Lateralization bugs refers specifically to the confusion between right turns and left turns. It may be thought that this would be a problem occurring at the lower grades but not at the grade 6 level. The findings show otherwise. Ninety-six instances of lateralization were noted. This accounted for 18.2% of the bugs. Although all of the students involved were aware of which hand was right and which was left, not all were sure which turn was needed when considering the turtle's orientation. One student explained that initially she had understood that the turn for the turtle was always the opposite of what it was for her in front of the screen, and she had just learned that it depended on the turtle's heading and not her position. It must also be noted that many of these bugs were the result of thoughtlessness and were quickly corrected. One student told the researcher that she thought most students put right for left because right was more common (a right handed world), so these bugs were tallied in two groups. In fact, there were 52 instances of RT for LT and 44 instances of LT for RT. Seven students erred on the side of having more RT for LT bugs while 6 had more LT for RT bugs. Four students had the same number of RTs for LTs and LTs for RTs, while 3 students were not observed making lateralization errors.

In School C, where the students had access to the computer lab and more computer time, and the programs were generally more advanced, only 11% of the total number of bugs were lateralization bugs. In School A, 19% were lateralization bugs, and in School B, 25% were lateralization bugs. These bugs appear to become less frequent with

more experience even though the task is more complex. In most cases they were easily corrected. Only when the lateralization was not identified as the cause of the problem was there difficulty.

Locating Origin Bugs

Bugs classified under locating origin represented 9.1% or 48 of the total bugs. Locating origin was defined to include the bugs occurring when a procedure or subprocedure was not started in the HOME or DRAW position at the center of the screen. The bug may come at the beginning of the procedure, or where two subprocedures are joined together. These bugs, of course, related to dimension and orientation bugs, since distance and angle (or position and heading) were often involved. The bug was classed as a locating origin bug when it appeared to be more than just a dimension or orientation problem, but rather a combination or factors.

The majority of the origin bugs occurred in the work of students who were using modular programming. Thirteen of the 48 bugs resulted from trying to find a corner location or appropriate place to begin the drawing. Thirty-six of the bugs occurred when subprocedures were being joined or distinct parts of a procedure were being located. Several of these bugs occurred when students in School C were tessellating shapes, particularly the irregular hexagon. Many of the bugs were the result of a subprocedure beginning with a turn command. The turtle would be positioned and then the subprocedure called, only to have the turtle reorient itself before drawing. This was especially difficult to control if the turn appeared in the beginning

of a REPEAT statement. Some of the bugs resulted from difficulties with the SETPOS command.

Omission Bugs

Bugs of omission accounted for 14% of the total. Over half of these bugs were the omission of PENUP or PENDOWN or both. Other things counted as omissions included the failure to include a line or lines of the procedure when copying from notes, from immediate mode or from another part of the program being used as a pattern. If changes were intentionally made when copying these were not counted as omissions. The omission of the turn command or forward command in a REPEAT statement was also considered here when it was obvious that it was an omission and not just a misunderstanding of the REPEAT statement.

Excess Information Bugs

Excess information in the procedures was considered a bug only when it interfered with the result expected. A number of instances of excess information went undetected by the student. Examples include PU HOME PD when the turtle was already at HOME, PU turn PD, retracing a line or shape for no apparent reason, moving the turtle with PU and then not following the move with some action, and REPEAT 1 [. . .] when just the bracketed part is needed. Excess procedures were also written and then later abandoned. These were included as bugs only if they in fact resulted in bugs during their construction, and not because they were there as excess. Bugs noted in this category that did affect the resulting graphics included excess length and turn

commands, excess PU and/or PD commands, excess variables, or inputs. These bugs accounted for 2.7% of the total.

Syntax Bugs

The category of bugs classed as syntax accounted for 4.7% of the total bugs and included such things as spelling errors, obvious typing errors, missing spaces, SETX SETY or SETPOS format, or incorrect name for a procedure.

Miscellaneous Bugs

The remaining 7.4% of the bugs were categorized as miscellaneous. These bugs included such things as the misunderstanding of the negative coordinate values for the DOT command, problems in rotating or moving a shape that has a fixed position command such as HOME, SETX, SETY, or SETPOS in the procedure, pen color not visible on certain background colors, using TO and ED inside a procedure, errors in mathematical calculations performed with pencil, semantic problems with REPEAT, PU (expecting turtle to move forward to top of screen), and PR (expecting turtle to move over to the right), and a LOGO SYSTEM BUG.

Debugging Strategies Observed in Students' Work

The grade 6 students observed in this study demonstrated a variety of debugging strategies, with a heavy reliance on a few strategies. There were a total of 1206 strategies identified on the observations sheets. These were recorded under 36 headings, which were then grouped into 14 major categories. These major headings and sub-headings are listed below.

1. Immediate mode
2. Observe and Change
   - Missing lines
   - Lateralization
   - PENUP - PENDOWN
   - Length or degrees
3. Guess and Check
   - Location of bug, RIGHT, LEFT
   - Trial and error
   - Reasonable guess
   - More precise
   - Aha
4. Play Computer - Search Listing
   - Line by line
   - Compare listing and graphic
   - Compare notes to listing
   - Compare notes and graphic
5. Play Turtle - Physical Referent
   - Walk
   - Move body, arm, hand, finger
   - Ruler or pencil
6. CLEARSCREEN - Start Again
7. Delete - Start Again

8. Simplify
   - SETPOS SETX SETY
   - REPEAT
   - Subprocedures
9. Seek Information
   - Book or chart
   - Friend
   - Teacher
   - Researcher
10. Follow Pattern
11. Error Messages
12. Change Plan of Approach
13. Abandon or Change Task
14. Miscellaneous
   - Draw diagram
   - Make notes
   - Calculate numbers
   - Abandon or delete procedures or change order
   - Use <CONTROL> G, STOP, PRINT POS
   - Experiment with inputs for variables and color

The major strategies and their sub-headings will be discussed in this section, as well as the frequency of use of the strategy. Table 5 gives the number of times each strategy was used by each student. Samples of students' work illustrating the various debugging strategies are included and discussed in Appendix 9.

Table 5

Debugging Strategies Identified

Name	1	2	3	4	5	6	7	8	9	10	11	12	13	14	Total
Barry	5	12	30	1	10		10	1	2	7	2	1	2	6	89
Carol	1	7	25	3	15		5					2		2	60
Cindy	2	4	23	4	14		6	1	6	5	1	1		6	73
Connie	1	2	2	1	3					2				1	12
David	4	4	8		6		4	1	1		2	1		1	32
Don		3	14				2			2		1			22
Doug	1	3	4				6	2			1		1		18
Mark	8	5	51		5		2	1		2		1	1	3	79
Mike		6	57	3	4		1		1	2	3	1		2	80
Murray	2	3	14	6	6		1					1		7	40
Lana	7	21	1	4	1					2				3	39
Laura	3	1	19	4	5		1				1	2		5	41
Linda	7	20	5	2		5			1	8	1		1	2	52
Ray	16	13	44	1		7	4	1		4		2		7	99
Sandy	17	12	34	2	2	24	3	3		2	2	2		1	104
Shauna	11	6	23	2	8	9	7	1	2	4	2			4	79
Ted	3	1	36				3	2			1	1		6	53
Trevor	14	7	65			45	2	2	1	1				3	140
Troy	3	8	20	1	1		5		2		7	1		2	50
Vicky	4	8	19		1	2		3	1	1	2		1	2	44
Sch. A	14	35	106	9	48	0	33	5	9	16	6	6	3	16	306
Sch. B	27	56	147	19	21	5	5	1	2	14	5	5	2	22	331
Sch. C	68	55	241	6	12	87	24	12	6	12	14	6	1	25	569
Total	109	146	494	34	81	92	62	18	17	42	25	17	6	63	1206
Percent	9.0	12.1	41.0	2.8	6.7	7.6	5.1	1.5	1.4	3.5	2.1	1.4	0.5	5.2	100.0

Key to Debugging Strategies Identified

1. Immediate Mode
2. Observe and change
3. Guess and check
4. Play computer - check listing
5. Play turtle - physical referent
6. Clear screen - start again
7. Delete - start again
8. Simplify
9. Seek information
10. Follow pattern
11. Error messages
12. Change plan of approach
13. Abandon or change task
14. Miscellaneous

Immediate Mode

Since one criteria for the selection of students for the study was that they be able to work in procedural mode, it was first thought that work done in immediate mode would not be analyzed for bugs and debugging strategies. It was found however, that for a few students nearly all debugging took place in immediate mode, and only those bugs that resulted from improper entry of notes taken during immediate mode actually occurred in procedural mode. Therefore work done first in immediate mode as well as work done in procedural mode was analyzed. Exceptions to this were a few isolated instances when the researcher was unable to record all the changes made in a rapidly executed series of changes, where the immediate mode work was more of an exploratory nature, or where the student worked with a full graphics screen and the keyboard input was not visible.

As mentioned, for a few students immediate mode was sometimes used first as a strategy for working out the bugs as well as for planning, before entering any commands into procedural form. The majority of students, however, used immediate mode as a debugging strategy following the entry of part or all of a procedure. When a bug occurred or when the student was unsure of the next part of a procedure he/she resorted to immediate mode to work it out. A total of 109 instances of the immediate mode strategy were recorded, or 9.0% of the total debugging strategies.

The amount of time spent working in immediate mode varied greatly. In one instance the whole period was spent working in immediate mode and making notes in a notebook. In several instances

the majority of the period was spent working in immediate mode, then transferring notes to procedural mode. For many students only short segments of the period were spent working in immediate mode. A few students did virtually everything in procedural mode, and did not make use of this strategy more than once or twice during all of the times they were observed.

Observe and Change, Add or Delete

In 146 (12.1%) instances, students were aware, as soon as they executed the procedure, what the bug was and how to make the necessary changes. Just observing the graphic result was all that was necessary, and usually no other strategy was evident. In fact, on a couple of occasions the student reentered the editor and made the changes even before executing the procedure, saying, "Oh, I forgot ...". There was seldom any hesitation in locating the exact spot in the procedure, to insert, change, or delete, or any hesitation in knowing what to insert, change, or delete. These instances were then classified as observe and change and treated apart from those in the next section, where it appeared that the student was guessing.

The observe and change strategy was most often the one occurring when the student had a lateralization bug. It was also used in conjunction with PENUP and PENDOWN omissions.

Guess and Check

Guess and check was used in a variety of situations and with a wide range in accuracy. Included are the very accurate, educated guesses, often made to determine precise values, as well as the obvious trial and error or random like guesses. Included also are the

reasonable guesses and the "Aha", insightful instances. With the exception of 2 students this was the strategy used most often and it accounted for 41.0% of the total debugging strategies. The 2 students for whom it was not the most used strategy did much of their work in immediate mode and did use the observe and change strategy in a number of instances. Guess and check was often used repeatedly in order to remove one bug, and each of these tries was recorded.

In many cases this strategy was used to obtain a more suitable value for a length or turn command or for a very precise value, while many other cases were clearly trial and error attempts to eradicate a bug. Sometimes this strategy was used in actually locating the bug in a procedure, when it was not known exactly what was wrong. In a small number of cases students assumed that it might be an incorrect turn, left for right or right for left, but did not know and were guessing. If they had known the instances would have been classified as observe and change. Twelve of the 494 guess and check instances were also recorded as insight or what one might call the "Aha!" experience, when suddenly the student "knows" what it is. In fact, not all of these instances of insight did eradicate the bug.

Playing Computer or Playing Turtle

Playing computer usually refers to going step by step through the listing and either mentally or with pencil and paper, doing each command as the computer would do it. With much of Logo being graphic and focusing on the moving of the turtle, playing turtle involves the use of body movements or some physical referent to determine the orientation and movement.

Playing computer accounted for 2.8% of the total debugging strategies. Students were observed searching for the bug by working line by line through the listing, comparing the listing to notes, comparing the listing to the graphic either on the screen or on paper, or comparing the notes to the screen or paper graphic. Notes here were usually those made by the student while working at the computer, usually in immediate mode, or notes previously written out and brought to the computer.

Playing turtle involved the use of a physical referent such as a pencil or ruler, finger, hand, arm, or the body and accounted for 6.7% of the total debugging strategies, with the use of the fingers being the most common. On occasion one student actually left the computer, taking pencil and paper, and walked out the turtle's projected path. The physical referent used often related to determining orientation, and was usually used in conjunction with some other strategy, often guess and check.

Clear Screen and Start Again

This strategy relates closely to guess and check, but adds that extra facet of having a fresh start by clearing the screen while working in immediate mode. Clear screen and start again accounted for 7.6% of the total strategies. It was used predominantly by 2 students who accounted for 69 of the 92 uses and included the repeated use of procedures in immediate mode in order to decipher how to connect them.

Delete and Start Again

When a student did more than just correct the bug in a procedure by deleting the actual bug, but rather deleted several commands, parts

of a procedure, or whole procedures, it was recorded as a delete and start again strategy. The student wanted a fresh start rather than try to fix the bug that existed, and would sometimes delete back to the previous point at which the procedure had been tested and was known to be functioning correctly, or far enough back to be relatively sure that the bug would be deleted. This strategy accounted for about 5.1% of the total.

Simplify

On a few occasions students were observed simplifing the approach that they were using in order to get rid of the bug. This involved switching from the use of SETPOS, SETX, SETY, and REPEAT to a string of single commands, from the use of a subprocedure, and from the use of variables. The removal of a turn command at the beginning of a subprocedure was also seen as an attempt to simplify. Approximately 1.5% of the debugging strategies were classed in this category. Only one example was observed where a student simplified, worked out the bug, and then returned to a more complex level.

Seek Information

Seeking information accounted for 1.4% of the strategies. Each of the following strategies occurred only once; use a book, consult with a friend, and consult with the teacher. On many occasions questions were asked aloud and of the researcher. These were, if at all possible, treated as if the student was just thinking aloud and not responded to by the researcher. Eleven instances, or less than 1% of the total debugging strategies were recorded as seeking help from the researcher. These were cases where the researcher felt compelled

to respond to the student's very direct plea for assistance, as discussed in Chapter II.

Following a Pattern

Forty-two instances, or 3.5% of the total strategies, were noted as following a pattern. These instances were those where the student actually copied a sequence of lines or used values obtained from observing the pattern being established. These were considered apart from the idea that much of what the students do comes from patterning. For example, the use of FD 40 and FD 20 for the height and width of letters was not included.

Error Messages

Of the total debugging strategies, 2.8% involved the use of error messages provided by the computer. The error messages relate closely to the syntax bugs. Students usually note the message and make the necessary changes at once, but it was observed on more than one occasion that the student ignored the message when it was not preventing the continuation of a subprocedure, then later had to deal with it when executing the superprocedure.

Changing the Plan of Attack

This strategy was used 1.4% of the time, sometimes in conjunction with deleting. Students were not changing or simplifying the task but just tackling it from a different angle.

Abandon or Change Task

Only three instances of abandoning the task were observed. Two of these were tasks that had been set by the student, while one was a specific assignment given by the teacher. Three instances were noted

where the task was changed by the student. All of these were changes in student set tasks. Abandoning and changing the task accounted for 0.5% of the strategy uses.

No attempt was made to include what might be considered changes in the intent of the assignment, since such changes appeared to be unintentional and a result of not clearly understanding and interpreting the assignment. Also some assignments were very open to interpretation or suggested further exploration.

Miscellaneous Strategies

The number of strategies included in this section is small (5.2%), but some of them were especially effective. While most of the students had some strategy listed as miscellaneous, the majority of the uses were made by 6 students.

Mathematical calculations on a set of numbers was observed as a strategy on seven occasions. Admittedly, many more caluclations were done mentally by students throughout their work, which were not noted as a strategy.

Four instances of drawing a diagram were noted, while 23 instances of making notes were observed as a debugging strategy. Some of these instances involved making notes throughout the entire period, of all work done in immediate mode.

A number of abandoned procedures are evident in the disk files of the students. Some are there as a result of the misspelling of a procedure name and were not erased, while others were subprocedures or procedures which were tried and then deserted. Observing and changing

the order of the use of subprocedures and the location of the subprocedure within the superprocedure was also noted.

The use of the PRINT POS command by one student was a strategy used to determine the location of the turtle in immediate mode so that it could be directed to that position within the procedure. Using <CONTROL> G and STOP to help locate a bug were strategies used by 2 students.

Experimenting, mainly with various background and pen colors, was done by several students. One session was also spent experimenting with cirle construction, and one with recursion. When unsure of the necessary inputs to a procedure, 3 students were observed using variable inputs and then experimenting until the necessary inputs were known. Two periods were spent by a student who was experimenting with a FAN procedure provided by the teacher.

How Strategies Relate To Bugs

More than one strategy was often used to eradicate a particular bug. This is illustrated in the last part of Appendix 9. Further evidence for this is obtained by comparing the total number of bugs found and the total number of strategies used. While there were 528 bugs recorded, there were 1206 strategies or 2.3 times as many. This is the result of two factors. On some occasions two or more strategies were used together on a single bug. This was often the case with immediate mode and guess and check; physical referent and guess and check; physical referent, guess and check, and immediate mode; and immediate mode, clear screen and guess and check. The

second factor resulting in a higher number of strategies being used than bugs occurring was the need on many occasions to make more than one attempt to eradicate a single bug. This was done by using the same strategy repeatedly or using various combinations of strategies on successive attempts. Table 6 shows the relationship of bugs to debugging strategies.

Since guess and check is the most used strategy accounting for 494 of the 1206 strategy uses, or 41%, it is not surprising that it is the predominant strategy used for 4 of the 8 classes of bugs. The 4 classes of bugs not having guess and check as the predominant strategy for removal are lateralization, omission, syntax and miscellaneous. Not counting the 17 miscellaneous strategies used on bugs classed as miscellaneous, there were 15 uses of guess and check and 15 uses of delete as the predominant strategy in dealing with miscellaneous bugs.

Lateralization bugs were corrected more than 52% of the time by just observing and changing. Guess and check, immediate mode and physical referent also were used repeatedly. Lateralization bugs were usually corrected on the first attempt, since the 96 bugs required only 115 strategy uses.

Omission bugs, which usually involved PU and/or PD, were also often solved by the observe and change strategy. Guess and check, checking line by line through the listing, and physical referent also played an important role in the correction of omission bugs. The 74 omission bugs required 106 attempts at correction.

Table 6

Relationship Between Type of Bugs and Debugging Strategies Used to Eradicate Each Bug

	Kind of Bugs								
Strategy	1	2	3	4	5	6	7	8	Total
Imm. mode	23	12	12	46	4		1	11	109
Obs-ch	15	15	60	2	45	1	3	5	146
Guess-ch	203	91	22	124	21	13	5	15	494
List	11	4	2	2	12		1	2	34
Phy. ref	21	21	10	16	11			2	81
Clear Sc	3	13	2	72				2	92
Delete	8	8	1	14	3	9	4	15	62
Simplify	1	2	1	8			2	4	18
Seek info	2		1	4	2	1	1	6	17
Pattern	24	5	1	2	2	2		6	42
Err mess					1	4	16	4	25
Ch. plan	2	1		9	1	1		3	17
Aband-ch	2			1				3	6
Miscell	14	10	3	14	4	1		17	63
Total	329	182	115	314	106	32	33	95	1206
Total Bugs	165	67	96	48	74	14	25	39	528

Key to Kind of Bugs

1. Dimension
2. Orientation
3. Lateralization
4. Locating Origin
5. Omissions
6. Excess
7. Syntax
8. Miscellaneous

The predominate strategy for correcting syntax errors was the error message provided by the computer, although guess and check or observe and change and delete were also used. Delete refers to the deletion of the line or section containing the bug and not just the correction of the actual bug.

The 48 locating origin bugs accounted for 314 strategy uses in attempts to eradicate them, or 6.5 times more strategies were used than bugs recorded. These were the most difficult of the bugs observed to deal with and instances of 7, 8, 9, 10, and even 16 tries, some with multiple strategy uses, to eradicate one orientation bug were noted. (This area seems to have the most unsolved bugs, although it is difficult to determine, since many projects were not completed during the research observation time.) The origin bugs also drew a heavy use of immediate mode, guess and check, and clear screen for a fresh start strategies. Eight of the 18 uses of simplify and 9 of the 17 uses of change plans also related to origin bugs.

While some bugs were mainly dealt with by using one strategy, others were handled in numerous ways. Also, some strategies were used mainly for certain bugs while others received very wide usage. The vast majority of all bugs were dealt with until eradication was achieved.

Attitude

The attitude of students while working on computers appeared to be very positive. Students came willingly to work at the computer. Often three or four students would meet the researcher at the beginning of the morning or afternoon, asking to be "first". Several students asked to work through their recess or into the noon hour period, and on a number or occasions students willingly stayed to work on the computer while classmates went to the library, music or phyical education class. These classes are often ones that students do not wish to miss, but a turn at the computer was much desired as well.

Don was the only student who at any time indicated that he did not wish to come to the computer for a turn. As soon as he was assured that he did not need to come, he was there, settled in within a very few minutes and had a profitable turn. He requested a second turn later in the day and this was granted.

Connie, who was a new member of the class and fairly new to Logo, appeared at times to lack confidence and sought reassurance. Vicki also was a somewhat reluctant worker. Both of these girls were pleased to be able to work at the computer, but are cited here as being less keen than the majority of the students.

The other 16 students portrayed a considerable amount of interest, eagerness, and pleasure in being able to work at the computer. This showed in their desire to have a turn, the enthusiasm with which they actually worked, and the pleasure they expressed as they completed the task or a part of the task. Examples of

observation and log notes are included in Appendix 10 to illustrate the interest of the students.

Attitude Scale - Debugging Computer Programs

All students in the study completed the 24 item attitude scale patterned after the "School Subjects Attitude Scales" by Nyberg and Clarke (1979) found in Appendix 2. Students chose between the bipolar pairs, and could obtain scores ranging from +48 to -48. Item 1 on the scale follows as an example.

worthwhile ___ ___ ___ ___ ___ worthless

Checking the first blank the student would score a +2, or a -2 by checking the last blank. The second blank would rate a +1, while the second last blank would rate a -1. The middle blank, a zero, would not affect the score.

This scale contains three sub-scales; evaluation (happy, lively, like, fair, interesting, active, enjoyable, favorite) was signified by a positive score, usefulness (worthwhile, student centered, practical, experiment, valuable, important, good, necessary) was signified by a positive score, and difficulty (not secure, strange, hard, failure, slow, unsure, tense, abstract) was signified by a negative score. Each sub-scale has eight items for a possible score range of +16 to -16. The scores from the 20 students are depicted in the stem and leaf charts in Tables 7 and 8.

Table 7

Attitude Scores - Total Score

4	55
4	3
3	8
3	33
2	789
2	24
1	5889
1	3
0	69
0	4
-0	2

Stem -tens
Leaf -units

Table 8

Attitude Scores - Subscales

	Evaluation	Usefulness	Difficulty
1	66 455	66 444	5
1	23 00	22 00000111	333 11
0	89 6777	9 7	9 7
0	3	22	44444 3
-0	001 2	1	1 21
-0			4 65

Stem - tens
Leaf - units

On the total scale, one quarter of the students scored at 33 or above, with the two top scores being 45. There were one quarter of the scores below 14, with the lowest score -2. The mean was 23.35. The usefulness scale received the highest score with the mean being 10.1, while the evaluation scale had a mean of 8.35. The mean on the difficulty scale was 4.9, with scores dropping as low as -6.

When considering the total score, the debugging of computer programs received a strong positive rating by the students. Only one negative score occurred among the 20 students. The 4 students who gave the lowest ratings on the total scale were David, Connie, Troy, and Doug. Doug rated debugging low on all three sub-scales, while Troy's name appears in the bottom four on all but the difficulty sub-scale. Connie's name appears in the bottom four on evaluation and difficulty, but is among the top four on usefulness. David's name is among the bottom four only in the usefulness sub-scale. The other scores in the bottom four on the sub-scales appear on only one sub-scale and not in the total. They are Carol's in evaluation, and Lana's and Linda's in difficulty.

The 4 students who gave the highest ratings on the total scale were Cindy, Mark, Ray and Murray. Cindy and Mark were among the top four on all three sub-scales. Ray was among the top four on usefulness and difficulty, while Murray was among the top four on evaluation. Connie, was also among the top four on usefulness, while Barry was the other student among the top four only on the difficulty sub-scale. (It must be remembered that having a high score on the

difficulty scale, means that it was considered less difficult, since this scale is associated with a negative rating.)

Those students whose names appear in the bottom four on any of the scales (Carol, Connie, David, Doug, Linda, Lana, and Troy) are those the researcher would describe as being the least advanced in their Logo programming, with the exceptions of Carol and Troy. Troy was able to handle subprocedures nicely and was using variables.

Of those students whose names appear in the top four on any of the scales (Cindy, Mark, Ray, Murray, Barry, Connie, Laura), Barry, Connie, and Laura would not be considered especially strong or advanced programmers. Connie was a recent beginner, and may with more experience do quite nicely.

Final Interviews

During the last visit of the researcher to each school, students were asked to respond to a number of ideas. Questions were read to the students from the interview form found in Appendix 3. Students were able to refer to the form or have questions reread if needed. Their oral responses were tape recorded and are printed and discussed here. For the first four questions students were expected to select one response, but some chose two. Second and third choices were encouraged on question five but not all students gave three choices. Thus the percentage totals do not all equal 100.

1. Completing a task (problem or procedure) that I start is
   a. very important - 40%
   b. important - 60%
   c. not important 0%
   d. other - 0%

2. When I have difficulty doing a task I feel
   a. challenged - 75%
   b. unsure - 25%
   c. discouraged - 10%
   d. other 0%

3. When I make errors I
   a. want to correct them immediately - 85%
   b. want to come back to them later - 15%
   c. never want to correct them 0%
   d. other - 0%

4. When I have all the bugs out of my work I feel
   a. satisfied - 35%
   b. relieved but want easier work next time - 5%
   c. encouraged to try something harder - 65%
   d. other - 0%

5. When I have difficulty doing a task I

	1st choice	2nd choice	3rd choice
a. quit	- 0%	0%	0%
b. try another approach	- 55%	5%	5%
c. look for materials to use	- 5%	20%	0%
d. ask a classmate for help	- 25%	35%	10%
e. ask a teacher for help	- 15%	25%	10%
f. look at other programs	- 0%	5%	0%

Students appear to be very committed to their task when they work on the computer. Completing the task was considered important to Shauna because "You learn more if you complete it. If you just leave it you won't learn anything." In response to question 2, David replied, "I'm gonna go ahead and try it again, and just keep on trying till I get it!" When considering the 81 observation sessions, where only three tasks were recorded as being abandoned, and the comments and choices made by the students, it appears that there is a strong committment to completing the task and working out the bugs that are there. There is also a desire by the majority to work out the difficulties on their own by trying another approach to the problem.

David's comment was, "I'd try it myself and then if it's really bad, I'd ask for help."

The errors or bugs that occurred in the work of the students interviewed are seen by most as a challenge, which they take up immediately. Many of them are encouraged to try something harder when they accomplish the task of removing the bugs from their program. Only 1 student indicated that she would want easier work next time, and no one wished to leave errors uncorrected.

How Do You Feel About Bugs? This question was asked of each of the 20 students interviewed? Their answers were varied and interesting. They are recorded here.

Cathy: It can be confusing, but if you really try you can usually fix them.

Cindy: Well, actually, I like fixing up all the mistakes I do because I'm learning a lot. Sometimes it's good to make mistakes cause if you've always got someone to tell you everything then you never learn. Most of the time they say you learn by your mistakes. (Researcher: Do you believe that's true?) Yip!

Connie: I don't really like them cause you have to go back and you have to figure everything out and it takes longer.

Barry: I don't like them that much. Sometimes they can be good - like they go and make more things for you. They make something that you didn't want and when you see what it is it looks a little better. Sometimes they go around and screw up your whole program.

Don: I don't like them. If you think of a real good program and then you get a bug in it, it doesn't turn out right. (But you can fix it, can you?) Ya. (How do you feel then?) Relieved.

David: I like to get it out and then the program is good. You get the bug out and it's okay then. Then you can do whatever you want.

Doug: I look forward to getting them out.

Mark: They're okay. Everybody makes mistakes.

Mike: Not very good.

Murray: I kinda feel frustrated but at the end I feel good if I can fix it.

Lana: I don't like them. I don't like them cause they make you have hardly any time to go on with your work. But then in another way they are good cause it makes - helps you learn when you have to do it over again.

Laura: Sometime they can be hard and sometimes they can be easy - like in the LOVE program that I did, it was kind of hard cause I couldn't figure out what was happening to my L or what was happening to my E. It just got me so confused. Then once you look at it again and try to figure it out it just comes to be getting easy. After it's all done I feel good.

Linda: I get kinda mad - cause, well, like, well, I don't know, I just get mad and go back into my program and see what the problem is.

Ray: It takes you awhile.

Sandy: They're okay! They're not bad when you figure them out.

Shauna: If they are interesting (make an interesting design) you figure out how they do it.

Ted: I don't really like them. When you are doing a program and you think you are going to have it done, when you run your program and it doesn't work, then you have to spend some more time and get the bug out.

Trevor: They are a little bit frustrating at times - when you have a project to do or something they get frustrating at times.

Troy: I don't think about them much, I just try to get them out - try to make them - not there any more. I don't think about them much.

Vicki: I don't like them.

Learning More About Logo. All of the students in the study indicated that they would like to know more about Logo. Some of the things they were interested in learning were: how to make designs and patterns, angles, shapes, circles, the editor, other commands, gaining more control, SETPOS, how to fill a shape, how to make our own procedures, how to make everything work better, how to program like our teacher does, and just everything.

The students had several ideas as to how they could learn more. Many took for granted that they were learning in school from the teacher, while others specifically mentioned the teacher. The following information resulted from tallying the responses of the 20

students. They were asked, "How can you find out more?" Their responses are recorded below.

Source	# of Responses
Teachers	11
Books	11
Experimenting	8
Classes & Camps	5
Friends	4
Someone who knows	3
Parent	1

Many of the students held the teacher in very high esteem. They wished to program like "teacher does". One said, "(Teacher) knows alot! He hasn't got around to teaching us all of it. We can't spend all of our time on computers."

Bugs and Causes. Students were asked in the interview to identify their most common bug and then tell what caused most of their bugs. The results of their answers are discussed here. Students' responses did not match very closely with the tabulated observations of the researcher. Some students may have been referring to their most difficult rather than most common bug, or those uppermost in their minds due to recency. The dimension bug was the most common bug identified for 14 of the students. (For 2 of these it was tied in frequency with another bug.). Only 2 students thought the dimension bug was their most common bug. Six students thought lateralization was their most common bug, but for only 1 of the 6 did this actually agree with the findings. Five students thought orientation was their most common bug, but this was the case for only 1 student. The selections of the students did not correlate well with the findings of the observations. It is possible that bugs mentioned were ones which

were particularly frustrating, recent, or unique and possibly mentioned for that reason, while the dimension bug which was so common was expected and accepted.

Responses to, "What is it that causes most of your bugs?" included:

"The commands you give; like you might say LT when it should be RT."

"Left Right bugs - I used to always think that it was the opposite way - somebody told me that. Like on the screen it was different - the right would be here and the left would be here instead of the left here and the right here. It depends on which way your turtle is facing. When we look at the screen this is our right and our left, but when you're facing this way, it's different."

"I had to move them. I wasn't sure how to do it so I just put that and after I saw what I had done I had to go back and put some moves in there."

"Not being careful."

"PU, PD - forgetful. Forward - you just don't know how far to go, and when you turn you don't know which way to turn sometimes."

"I just get mixed up."

"I'm not sure cause I don't write it down. I just go straight to the procedures. I don't really . . . plan ahead cause if I plan ahead I just can't imagine it on the screen."

"When you're doing the stuff on the computer (immediate mode) under the screen and writing it in your paper, sometimes you don't write all the procedure and you forget something and it just misses it

out. It's more fun to type on the computer than to write it down in your book. People think, 'Oh, I'll remember it all,' and then when it comes to doing it in your procedure you forgot if you wrote it down."

"Carelessness. Sometimes I just forget and sometimes I don't mark it down in my book."

"I just get nervous and I forget what I'm doing and I write one and it's not right so then I write the other one." (LT or RT)

"You copy wrong." (From immediate mode.)

"Not paying too much attention - just typing - like you think you know what it is but just typing you really don't."

"You're not putting the right angle - the right turn - left and right and 40 and 50."

"Just not knowing how big it has to be. Figuring out how big to have each shape and have it fit the screen."

"I need to experiment and that causes bugs - which way to go." (Getting a plan.)

Strategies. Students were asked, "When you have bugs in your work what do you most often do to get rid of them?" The list on the student interview form, which appears in Appendix 3, was provided for students to observe and comment on, or they could just discuss those strategies which came to mind. Some of the strategies which received a high number of student responses but were seldom seen by the researcher, such as ask teacher or friend, write notes, and draw a diagram, may have been used more by students if they had not been in the direct observation situation of this project. Their responses are summarized in Table 9.

Seventeen of the 20 students indicated that they made use of guess and check as a strategy for getting rid of bugs. Fourteen students said they write notes, 14 use their fingers, hands, or pencils, and 14 follow line by line through the listing comparing it with the resulting graphic. Ten students delete, usually just a part of the procedure, and start over, and 10 students said they asked a teacher or a friend, although many indicated that that would not be their first approach. Drawing a diagram was mentioned by 9 students;

Table 9

Strategies Students Say They Use

17*	Guess and check	8	Calculate numbers
14	Write notes	8	Consult book or chart
14	Use finger, hand, pencil	8	Follow line by line - notes
14	Follow lines - graphic	4	Play turtle
10	Delete and start over	3	Modular programming
10	Ask teacher or friend	4	Experiment**
9	Draw a diagram	2	Immediate mode**
8	Look for a pattern		

* Number of students out of 20 who said they used that strategy.
** Were not on original list. Likely reason for the few choices.

looking for a pattern, calculating numbers, consulting a book or chart, and comparing the listing to student notes were each mentioned by 8 students. Four students indicated that they play turtle, while 3 said that modular programming was a help in debugging. Four students mentioned experimenting and 2 mentioned working out bugs in immediate

mode. Neither of these items were on the list provided. Had they been they likely would have been chosen by a larger number of students. The pencil turtle - turn chart item was not selected as these were not available to the students in the study, although they had been observed in the pilot study school.

The term modular programming was not known to many of the students and many did not use this style of programming. If students asked about the term on the list, it was explained. Cindy commented on the usefulness of modular programming by saying, "That's what I'm doing with my FACE (the name of her program). Ya, it (helps to get rid of bugs) cause instead of just calling up the whole program ... if you break it up into procedures like that it's a lot easier. You know, I've got it all procedured out - right? Instead a lot of people, what they do is just make it one long one and put everything in it and if you got mixed up and you don't name it when you're going to have to go through your whole program fingering it out and figuring it all out. So it gets really confusing, but if you have different procedures and stick them together you don't have to do as much, well, it's just easier."

Other Considerations

The topics discussed in this section were noted as the study progressed. Although they are not necessarily bugs or debugging strategies, they relate to bugs and debugging and often have a bearing on the effectiveness of the students as they program.

Mode of Development

The 81 observations of the 20 students in the study were classified as to the mode, immediate or procedural, in which the development took place. Two students did all of their development in immediate mode and then transfered their work to procedural mode. There were also 2 students who did all of their developmental work in procedural mode. Six students did all or mostly all of the developmental work in every observation in procedural mode. When considering the total number of observations, 43% of them were done totally in procedural mode and 67% were done mostly or totally in procedural mode, while only 14% were done totally in immediate mode and 22% done mostly or totally in immediate mode.

Table 10 depicts the distribution of the observations according to mode. This information can be related to specific students in the study by examining the data in Appendix 5.

Table 10

Immediate Mode Verses Procedural Mode for Program Development

Time	Number of Observations Immediate Mode	Procedural Mode
All	11	35
Most	7	19
Half	9	9

Fifty-three of the sessions began by the student working in procedural mode. Twenty sessions were started in immediate mode, while three sessions were started with the student doing some experimenting. There were five sessions where the student began by debugging work from the last session.

Procedures and Subprocedures

Although some developmental work was done in immediate mode all students were writing procedures, if not to develop their work, at least to retain their work. In only one of the 81 observation periods was there no procedure written, added to, or debugged, and that period was the first short period in which Murray worked only in immediate mode, making notes. He entered the procedure during the second observation period.

Seven of the 20 students did not use subprocedures at any time during the observations. The complexity of superprocedures and subprocedures varied considerably. Cindy wrote a superprocedure calling eight subprocedures, while several students had superprocedures calling multiple subprocedures. A variety of techniques were used for joining procedures. Cindy had one superprocedure which called the eight subprocedures in turn.

```
TO MJ
PU
LT 90
FD 24
PD
REPEAT 18 [FD 30 RT 30]
JA
PU
FD 20
PD
JA2
NOSE
MOUTH
EAR1
EAR2
VG
TEETH1
END
```

Kelly's superprocedure calls three procedures, two of which call other subprocedures. The structure of his program looks like this:

```
ALL
    - REC6 -  MOVE
           -  REC5
           -  REC2 - used 4 times

    - REC7 -  REC2

    - MOVE2
```

Trevor's program, shown in Figure 6, is an example of the way in which some of the procedures were chained together, with each new procedure written calling the one before it.

Figure 6

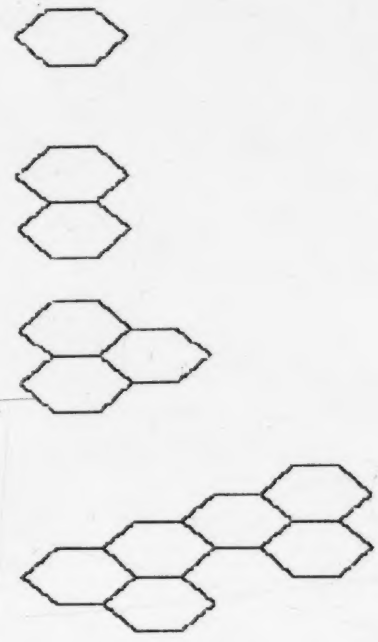
Trevor's Hexagon Tessellation

```
TO SHAPE
RT 90 FD 20 RT 45 FD 20 RT 45 FD 20 RT 45 FD 20 RT 90 FD 20
END

TO T
SHAPE
RT 45
FD 20
RT 90
SHAPE
END

TO S
T
BK 20
RT 45
RT 45
RT 45
SHAPE
END

TO Y
S
RT 45
FD 20
LT 45
S
END
```



Ray's first session was a beautiful example of top-down structured programming. After checking the location of SETPOS [-130 -113], he began:

```
TO C        (Started to put CASTLE but was not sure
FRAME       how to spell it.)
BUILD
BUILD2
BUILD3
BUILD4
DOOR
END
```

He said, "Do that for now," and then proceeded to write procedures for FRAME and then BUILD. When Ray came back for the next session he continued on, but rather that call the superprocedure to test his work as he went, he chained the procedures together in a manner like that mentioned above. When questioned about this he explained that he liked to do it that way (chaining) until he was finished and then he would remove the procedure names and call the superprocedure. He did not like to have the messages saying, "I DO NOT KNOW HOW TO DOOR (or the next procedure undefined)" appearing on the screen.

Meaningful names for procedures and subprocedures appeared to be important to some students but not to all. This did not appear to be an important factor in debugging procedures, as the students were not working with a lot of procedures at any one time. It is interesting to note, that of those students who used four or more subprocedures, all but two used meaningful names for their procedures. The two exceptions did use some meaningful names but not all. This can be observed in Appendix 5.

The testing and debugging of procedures was frequent and ongoing. Students did not usually develop more that a few lines at a time without testing and debugging. This was true not only for those who worked with long linear procedures but also for those using subprocedures.

Preventative Debugging

Sometimes students engaged in activities that appeared to be measures taken to avoid the occurrence of a bug (preventative debugging). Testing a procedure a second time to determine the position (location and/or orientation) of the turtle in an attempt to avoid an interfacing or joining bug was an example of preventative debugging. This practice was noted on several occasions. Ray made use of PR POS to determine the location of the turtle so that the correct coordinates could be use in the SETPOS command. Many instances of using a physical referent such as the hand or arm for determining the correct orientation and avoiding a lateralization bug were observed. The physical referent was also used as a debugging strategy. It was first intended that immediate mode work be listed as preventative debugging, and in some cases it was, but considerable actual debugging as well as procedure development takes place in immediate mode.

Refining, Expanding, and Exploring

Refining, expanding, or further exploring after the task had been completed and the procedure worked, was not a common occurrence. A few of the students spent time exploring with their procedures by changing input values. This was the case several times with those

students who made use of color. Various background and pen color combinations were tried. A few instances of refining also occurred. REPEAT statements were used to replace a number of single command lines and the replacement of repeated lines with a subprocedure were observed. Two of the students used a process of editing out the RETURN key strokes to draw the string of commands into one line to be included in the REPEAT statement.

Off-Computer Debugging

Throughout the study the majority of the students did not work on the computer or on their programs except when being observed. In School B it was apparent that the students discussed their computer projects with each other, and shared ideas about possible projects. The commands that Mike used to make the heart shape were shared among other students, but Mike was able to handle them well, doing the necessary debugging and making size adjustments to meet his needs. When Murray came to the computer for one of his sessions he indicated that he had thought about the problem he had making the letter N and was going to try it a different way. He did have some difficulty with his new approach as well and commented, "I don't really know what I am doing. I should have written it down." Sandy started a session without loading her file from the previous day. She wished to have a fresh start and made a regular hexagon instead of the irregular one she had previously used. Trevor also indicated on the last day when the interviews were held, that he had a new approach to try on his hexagon procedures. Some discussion had gone on among the students

in School C and they were attempting to assist each other in the process of debugging.

Number of Tries Per Bug

Not every bug is easily removed even after it has been identified. Many bugs required multiple attempts and often several different strategies in order to be successfully removed. This is evidenced by the fact that 528 bugs were recorded and 1206 strategies. Lateralization bugs, penup/pendown omission bugs, and some syntax bugs were among the easiest to correct, while those of locating the origin point, especially when interfacing two procedures or subprocedures, often required many attempts, and much more time.

Time Verses Accomplishment

The amount of work accomplished was obviously dependent on a number of factors, some of which are; the difficulty of the task, the experience of the programmer, the knowledge of the programmer, and the style and aggressiveness of the programmer. The work of 3 students is included Figures 7, 8 and 9 and discussed in this section to illustrate the variety in accomplishments. All three of these students were considered to be average or slightly above in their academic work, and they all have above average scores on the intellegence tests. Carol had been exposed to Logo over the longest period of time and Connie the shortest. The first is the work of Connie who was relatively inexperienced and somewhat reluctant. She worked on her project for 30 minutes. Connie worked on these procedures again during the next period, putting the three of them into a superprocedure. The next is the work of Ray who is a quiet,

confident, fairly aggressive worker. He spent 35 minutes on his project. Ray worked on this project again the next day, completing the dots across the front and adding a door. Last is the work of Carol who also appears quiet and confident but not aggressive. She worked for 40 minutes on her project. Carol also returned to her project the next session, removing the bug and making her design into two symmetrical sets of three squares each.

Figure 7

Connie's Work - Three Procedures - 30 Minutes

```
TO FOOTBALL
REPEAT 4 [FD 40 RT 90]
END

TO TE
FD 50 RT 90 FD 25 BK 50
END

TO YOU
REPEAT 2 [FD 60 RT 90 FD 90 RT 90]
END
```

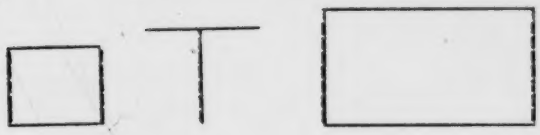
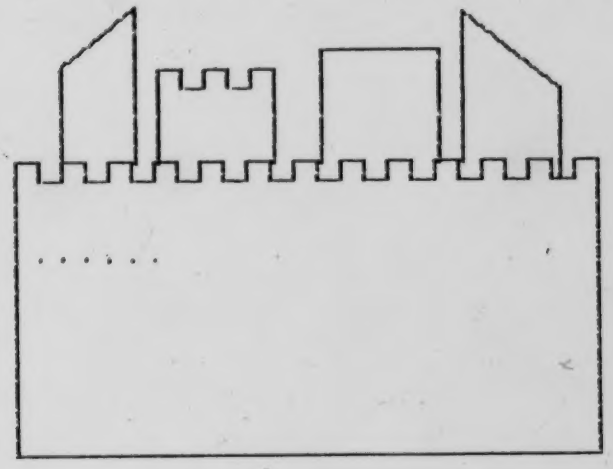


Figure 8

Ray's Work - A Castle - 35 Minutes

```
TO C
FRAME
BUILD
BUILD2
BUILD3
BUILD4
DOOR
END
```



```
TO FRAME
PU SETPOS [-130 -113]
PD FD 150
REPEAT 12 [RT 90 FD 10 RT 90 FD 10 LT 90 FD 10 LT 90 FD 10]
RT 90 FD 10 RT 90 FD 150 RT 90 FD 250
RT 90
END

TO BUILD
FRAME
FD 150 RT 90 PU FD 20 PD LT 90 FD 50
RT 45 FD 45 RT 135 FD 82 LT 90 PU FD 10 RT 90
END

TO BUILD2
BUILD
PD LT 180 FD 50
REPEAT 2 [RT 90 FD 10 RT 90 FD 10 LT 90 FD 10 LT 90 FD 10]
RT 90 FD 10 RT 90 FD 50
END

TO BUILD3
BUILD2
LT 90 PU FD 20 PD LT 90
FD 60 RT 90 FD 50 RT 90 FD 60
END
```

Figure 8 (Continued)

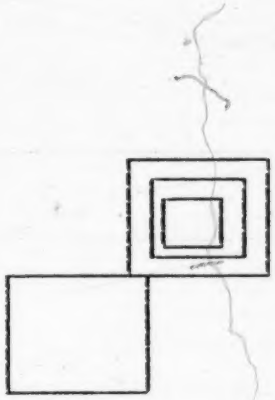
```
TO BUILD4
BUILD3
LT 90 PU FD 10 LT 90 PD FD 80 RT 135 FD 60 RT 45 FD 46
END

TO DOTS
PU SETPOS [-130 -113] LT 180
FD 100 PD
DOT [-120 -13]
DOT [-110 -13]
DOT [-100 -13]
DOT [-90 -13]
DOT [-80 -13]
DOT [-70 -13]
END
```

Figure 9

Carol's Work - Symmetrical Squares - 40 Minutes

```
TO WAS
REPEAT 4 [FD 60 RT 90]
PU
RT 90
FD 10
LT 90
FD 10
PD
REPEAT 4 [FD 40 RT 90]
RT 90
FD 5
LT 90
PU
FD 5
PD
REPEAT 4 [FD 25 RT 90]
LT 90
LT 90
PU
FD 15
RT 90
FD 10
LT 90
RT 90
BK 3
PD
REPEAT 4 [FD 60 LT 90]
END
```



Bug Serendipity

On several occasions bugs resulted in very attractive designs that were not sought for as part of the task being done. These bring surprise and pleasure to the students, and sometimes distract them momentarily from the original task. These interesting, unexpected designs help add acceptability to errors that occur in the procedures students write. A few examples of the graphics displays, from the work of students tessellating hexagons and from Mike's recursive heart procedure, were included in Figure 5.

Bugs Verses Ability

The total number of bugs per student is not a very meaningful figure in most cases. There are several reasons for this. First, not all students were observed for the same length of time. Secondly, not all students worked at the same speed or accomplished the same amount. Therefore, a low number of bugs does not mean that the student is a superior programmer, nor does a high number of bugs mean that the student is a weak programmer. In fact, the converse is more likely to be true.

Styles of Programming Exhibited by Individuals

A variety of programming styles were observed. The work of some students will be mentioned briefly in this section.

Lana's work was representative of the linear or "spaghetti" style programmer. The whole procedure was one long string of commands, each on a separate line. The complete program was contained in the single procedure. Although she would do a short section and test that

section, Lana's final work showed no signs of modular or structured programming.

Linda's program consisted of a number of subprocedures, each called in succession by the superprocedure. Her subprocedures resembled the "spaghetti" program except that it was chopped into short lengths with the breaks coming at logical locations, such as the end of each letter in the formation of the word LOVE. In one program, Linda showed that she had some idea of effective modular programming when she repeated the use of one subprocedure in the superprocedure.

Mike used subprocedures and also used several multiple command lines. He showed skill in adjusting the order of commands within a procedure in order to avoid the restricting use of HOME.

Ray had a good grasp of Logo primitives and used them to his advantage. The commands within each line were grouped for clarity and procedures were named with thought. His programming was very easy to follow and showed a logical organization of subprocedures and superprocedures. In the first observation session Ray showed a very excellent example of a top down structured program.

Trevor spent some of his computer sessions experimenting. He liked to try things out and see what resulted. Other sessions were spent with a definite project in mind, during which time he demonstrated a clear understanding of modular programming. He also made use of top down structured programming.

The majority of Don's work could be referred to as relatively undirected, accidental programming. He would type in lines with

strange number combinations and multiple commands within REPEAT statements, and wait to see the result.

Troy made use of subprocedures and superpocedures. He sometimes used a chaining technique to join his procedures. For example, REC4 called REC3, which called REC2. This technique was seen in the work of several students.

Although the styles differed, all students encountered bugs. They had developed some skill with debugging strategies and worked at attaining a bug free program.

Other Difficulties Students Encountered

A number of difficulties were faced by students. Although these are not bugs they may have been the cause of bugs and did present debugging problems to many.

Having a turn command at the beginning of a subprocedure or as the first part of the REPEAT command was one of the most predominant difficulties. Students would get the turtle positioned and then call a subprocedure that started with a turn command only to have that part of the diagram at the wrong angle. Often students did not know why the join was not properly made. Sometimes they knew why, but had trouble compensating for the beginning turn. Not finishing off a shape with the final turn to achieve the total turtle trip also caused problems when joining subprocedures.

Many students found it difficult to determine the length of diagonal lines. Some also were unsure of the turns needed if the diagonal did not use a 45 degree turn.

The REPEAT command was misunderstood by a number of students. Several students would put a forward, a turn, and a forward within the brackets, when they really wanted just a forward and a turn, or wanted a forward, a turn, a forward, and a turn. Several students were not able to make a rectangle using the REPEAT command. Trying to insert too much within the REPEAT also made it difficult for students to understand what they had.

Another minor difficulty was faced when students tried to debug procedures with the turtle hidden. This was especially troublesome if the bug was the result of a missing PENDOWN command.

Some students faced problems in trying to interpret the assignment. Others had little idea of projects to try on their own when they had completed the assignment given by the teacher. Other students were pleased to have the freedom of choosing their own tasks.

Summary

This chapter provided an explanation of the data analysis procedures and information on the bugs observed and the debugging strategies used by grade 6 students. The strategies were related to the bugs and students opinions were reported. A discussion of students' attitudes and a number of other considerations followed.

CHAPTER IV

SUMMARY, FINDINGS AND IMPLICATIONS, CONCLUSIONS AND RECOMMENDATIONS

A summary of the investigation is presented in the first part of this chapter. The findings and their implications are discussed and related to the literature reviewed. This is followed by an enumeration of the conclusions drawn from the study. Finally, some recommendations are proposed for research and for teaching.

Summary

In the area of computer programming instruction in school, and specifically in relation to the use of Logo in the classroom, there is considerable focus on the importance of debugging Logo procedures as a valuable aid in the development of thinking skills. It is pertinent then that the process of debugging be understood.

The major purposes of this study were to:

1. identify and classify the bugs present in the work of students,

2. identify the strategies that students use to debug Logo programs, and

3. describe the attitudes of students toward bugs.

The data for this exploratory study of Logo programming bugs and debugging strategies exhibited by grade 6 students in 3 elementary schools in Edmonton were collected during January and February, 1985. The schools were chosen with the assistance of the Edmonton Public School computer consultants. Teachers and students who were actively

involved in Logo programming and willing to participate in the study were selected. Students were chosen in 2 of the schools by their teachers as being representative of the class in regard to Logo programming. In the third school all of the grade 6 students were observed in the split grade 5 and 6 classroom. A fairly wide range of programming ability was evident. Actual programming assignments were those given by the teacher as part of the ongoing program.

Each of the 20 students involved in the study was observed an average of 4 times for an average of 30 minutes each observation period; a total of 2 hours per student. In addition, each student was also involved in a preliminary acquainting session and culmination questionnaire and tape recorded interview sessions. During the observations, extensive notes were taken, disk files were kept, and tape recordings were made where possible. The notes taken included the actual input students were entering at the keyboard, the comments made and actions observed, and notations about attitudes evident. Log notes were also kept throughout and at the end of each day.

Discussion and Implications of Findings

Bugs

Among the 528 recorded bugs, a total of 25 types of bugs were identified and grouped into 8 categories. The categories, when listed in the order of frequency of bugs with the number of cases in parenthesis, are dimension (165), lateralization (96), omission (74), orientation (67), locating origin (48), syntax (25), excess (14), and there were 39 miscellaneous bugs.

The dimension bug, which accounted for 31.2% of the total bugs recorded, was the most prevalent bug for 14 of the students. (For 2 of these students the number of dimension bugs was the same as the number of bugs in another category.) Orientation bugs (12.7%) were also very prevalent. Since the students' work was graphic and much of it was without set dimensions or orientations, it was not surprizing that these bugs were common. Relatively little formal work in school has been done with degrees. It is surprising, therefore, that students do as well as they do in the area of orientation. The less regular the shape the more likely the incidence of orientation bugs.

The number of lateralization bugs (18.2%) was higher than expected by the researcher, considering that the students were in grade 6. It was evident that those students who were more experienced and engaged in more advanced programming tasks had fewer lateralization bugs. Small pencil turtles which indicate left and right would be useful for those students who have numerous lateralization bugs.

Locating origin bugs (9.1%) were very time consuming for students to remove. This was likely because a combination of factors was often involved including dimension, orientation, the semantics of the SETPOS command, and the logical thinking required. It was also noted that a turn command at the beginning of a subprocedure made the situation more complex. The difficulties experienced by children in this area support the writings of Solomon (1975, 1976), Watt (1979), and Gorman (1983).

Locating an appropriate origin presented students with the most difficulty. The majority of unsolved bugs was also related to origin. The provision of manipulative aids and charts would be of value in this area. Either individual student or wall charts of the Logo screen coordinate grid could be designed by students and used as a reference when needed.

Syntax bugs (4.7%) were not nearly as prevalent as the researcher had anticipated. Since students had not had keyboarding instruction many more typing errors were expected. Also the limited background in Logo might have resulted in more incorrectly formatted commands. It appeared that most students worked fairly carefully and avoided commands if they were unsure of the syntax.

Students often forgot to include PENUP and PENDOWN commands. One wonders if the reason for this frequent forgetting is related to the attention that is placed on obtaining the correct destination for the turtle. Whatever the cause and wherever the bug (at the beginning of the main procedure, within the procedure, or in a subprocedure) it still had to be dealt with by every student in the study.

Meaningful procedure names were used by some students. The encouragement of this practice is advisable, especially as students write programs with multiple subprocedures. Clean files were kept by a number of students, while others made no attempt to delete procedures no longer desired. These procedures were sometimes mistakenly called again, especially if the name was not descriptive.

Some bugs were easily corrected in only one attempt. This was true for the majority of the lateralization, omission, and syntax

bugs. Other bugs were much more difficult and time consuming to correct and involved several attempts and often a combination of strategies. Locating origin bugs, found in the work of those students tessellating shapes as well as other origin bugs, were especially difficult. Up to 16 attempts at correcting 1 bug were recorded, and in excess of 5 attempts per bug were evident on a number of occasions. Dimension and orientation bugs were frequent but many of these were easily corrected because they often needed only fine adjustments for a more precise design.

Debugging Strategies

The 1206 identified strategy uses were grouped under 14 headings. The strategies, in order of frequency, are guess and check (494), observe and change (146), immediate mode (109), clear screen and start again (92), play turtle - physical referent (81), delete and start again (62), follow pattern (42), play computer - search listing (34), error messages (25), simplify (18), seek information (17), change plan of approach (17), abandon or change task (6), and there were 63 miscellaneous debugging strategies.

A wide variety of strategies was used by some students while others used relatively few. Most of those students who had a high number of miscellaneous strategies were also using most of the other 13 identified strategies. It appeared that the programmers considered by their teachers and the researcher to be more advanced were also those using a wide variety of debugging strategies.

Using error messages effectively resulted only if students were familiar with the error messages, knew how to read them, and

understood what they meant. Error messages were received by several of the students. In many instances the message referred to syntax bugs, were understood, and necessary debugging took place. On some occasions the message was ignored or not understood and was not dealt with until no other alternatives existed. Since the work of some of the students was at a relatively simple level, they did not receive more than the simplest messages. The use of STEP, TRACE, WAIT, and PAUSE were suggested by Gorman (1983) and Nelson (1985) as effective debugging strategies. These were not used by students in the study although one student did use <CONTROL> G, one used STOP, and another asked about slowing the turtle down. The use of error messages and built in debugging tools may be more evident with more advanced Logo students, although Weintraub (1985) indicated that highschool students do not use the error messages fully in the debugging process. This may be an area in which some instruction is required.

Playing computer, or going step by step through the listing doing what the computer would do, as suggested by Bruce (1980), was a strategy used by 13 students in the study. Twelve of these students plus another 3 were also users of what has been called playing turtle, as suggested by Solomon (1975), or using a physical referent and following the movement of the turtle. The results of this study support the earlier research on the effectiveness of playing computer and playing turtle, but not in regard to the effective use of manipulative aids. Manipulative aids, suggested as a debugging device by Solomon, were not used by students in the study. It is assumed

that the students in the study did not have access to aids that could have been used for debugging purposes.

Desk checking for bugs, as suggested by Cassel and Swanson (1980) does not appear to be a first step in debugging for grade 6 Logo students. In only a very few instances did students come to the computer with programs written in advance. Most students worked directly at the keyboard, composing and constructing the program as they worked, either in procedural or immediate mode.

The use of immediate mode to work out difficulties was valuable to students, even when they did not take notes to transfer into procedural mode. The combined strategies of immediate mode and making notes could be recommended to students. For those students who do all of their program development in immediate mode some encouragement could be provided to use it less often. If used just for resolving difficulties or testing ideas progress could be speeded.

One fairly effective programmer in the study who was noted using top down structured programming on occasion, said that he didn't really plan ahead and write things down, but just went straight into writing procedures. Another said he pictured things in his head rather than drawing diagrams. The careful planning, writing, and checking on paper that Cassel and Swanson (1980) discuss is not an approach used by the grade 6 Logo students observed in this study. The statement by Leron (1985) that suggestions to plan ahead or look back are often met with resistance by children may explain the lack of planning ahead by students in the study and also the unneeded procedures or parts thereof which were not removed.

If students were required to do some preplanning during their off-computer time, they would likely benefit in two ways. First, the planning time would result in a more carefully structured procedure. Secondly, hands-on time would be used more efficiently for entering and debugging rather than planning.

Those errors appearing in the work of the students are limited by the type of assignments they were involved in and by the degree of sophistication and level of programming expertise the students had attained. Students working at more advanced levels would likely encounter some of the same bugs but would also encounter others such as those met in list handling, mathematical operations, conditional statements, interactive procedures, handling outputs, and file commands. One student referred to conditional statements but said that he was not sure how to use them. The student who had a recursion bug in his work was also not aware of how to use conditional statements.

Some strategies which may normally occur were not present or very rarely seen during the study. Only one instance each of asking a friend, asking the teacher, or using a book was recorded. This was likely due to the nature of the study and the fact that the researcher observed, took notes, and in some instances tape recorded in close proximity to the student. A number of instances of talking to or asking the researcher occurred, and though response and help were seldom provided, one would assume that talking through the problem may have been a helpful strategy. Bruce (1980) and others would agree that this is a useful debugging strategy.

Attitude

Responses to the adapted Nyberg and Clarke (1979) attitude scale indicated positive to very positive attitudes toward the debugging of Logo programs, with only one negative total scale score occurring. The highest scores occurred on the usefulness sub-scale, while the lowest scores were on the difficulty sub-scale.

The comments from the interviews with the 20 students, on how they feel about bugs, indicate that negative feelings do exist for some of these students. However, many of the students displayed and expressed a very positive attitude. The relationship between attitude, length of time the student has been involved with Logo, and the degree of understanding and success with Logo is not known.

A positive feeling of control and success cannot be expected to develop for the student if he is not successful in debugging the program. Those using multiple attempts to eradicate one bug often worked without showing signs of frustration, but some of them did express the concern that bugs take too much time. There were very few instances in this study where the student was not successful or at least partially so. Occasionally the student would adjust the task but usually they appeared to be satisfied with the results they obtained. Abandonment of the task occurred only three times.

Conclusions

The following general concluding statements are supported by the results of this study.

1. The Logo programming work of grade 6 students results in many and varied bugs.

2. The most predominate bug which occurred was dimension, including length of line or side of shape, distance between parts, number of REPEATs, or circle dimensions.

3. Lateralization (confusing left and right), omission and then orientation bugs were next in frequency, followed by locating origin, syntax, and excess.

4. Many and varied strategies were used by the students in order to rid their programs of bugs.

5. Guess and check was by far the most common strategy, accounting for 41% of the uses. Guesses varied from being very reasonable and accurate to being very unreasonable and random.

6. Rarely (0.5%) did students abandon or change the task as a strategy for dealing with bugs.

7. Multiple strategies were often used to correct the same bug.

8. Many bugs required more that one attempt at debugging before they were eradicated.

9. Lateralization and omission bugs were usually debugged by observing and changing.

10. The error message provided by the computer was usually sufficient to correct syntax errors.

11. Lateralization bugs were the easiest to debug, while locating origin bugs were the hardest.

12. Students attitudes regarding debugging ranged from positive to very positive. Bugs were disliked by some students, often because they were time consuming, but were seen by some as a way of learning things they might otherwise miss.

13. Students stated as well as illustrated their commitment to the completion of tasks and the challenge of debugging.

14. Little evidence of detailed planning prior to keyboard work or looking back to refine a procedure or make it more efficent was noted.

15. Assignments given to students were fairly open ended and provided for the needs of most students. There was a wide range of results produced by the students.

Recommendations for Research

This study was exploratory in nature and therefore limited in scope. It would be desirable to observe and compare the results when a more tightly controlled classification of student ability, knowledge of Logo, and experience with Logo was used.

Some students made use of a wide variety of strategies. A heavy reliance on guess and check existed for most students. A study in which an attempt to teach strategy use and application for debugging may provide further insights into how strategies are learned and applied.

No attempt in this study was made to influence the type of assignments in which the students were involved. Most assignments were very open ended, although some students were provided with written instructions and some guidance. Further research is needed into the effect of the amount of structure and guidance provided in the assignment and expectations and level of achievement required and their relation to attainment, growth in Logo knowledge and expertise, and improved application of debugging strategies.

Different programming styles were observed during the study. Many of the students using a modular approach to programming were accomplishing more and tackling more difficult tasks, but appeared to encounter the same kinds of bugs as non-modular programmers and with a fairly high frequency. Further research might be conducted into the relationship between programming style and the kinds and numbers of bugs which occur. Those styles most effective could then be encouraged.

Lateralization bugs were much less frequent among the students in one school. A student there indicated that she had previously made use of a physical referent and doesn't have much trouble with lateralization bugs now. A longitudinal study into the bugs and strategies which consistently remain over time, those which become extinct, and the process of extinction would be helpful.

Recommendations for Teaching

Some recommendations were alluded to in the discussion section of this chapter. In addition some more general recommendations for teaching can be made.

The evident lack of effective strategies besides guess and check for some students suggests a need to search for helpful strategies to add to students' repertoire. Students could be assisted to use guess and check as efficiently as possible by providing companion strategies such as following patterns, using physical referents, manipulative aids, charts, diagrams, graph paper, and note taking. It is recommended that students be given help in developing new and effective strategies for debugging.

Since STEP, TRACE, WAIT, and PAUSE are considered to be important debugging strategies by Logo researchers, yet were not evident in the work of any of the students in the study, it must be assumed that these strategies need to be taught. Beginning Logo students have many new ideas to master and would not be aware of these strategies or their value. Some of the students in the study showed a need for these strategies. Teachers need to make available to the students information about these strategies and an opportunity to practice using these and other related strategies.

The provision of additional guidance for students who are uncertain about an assignment or show difficulty handling very open ended assignments could provide for growth and encouragement for those students. Teachers can extend learnings and encourage students to try

new approaches and encounter new Logo situations to strengthen thinking skills. Patterns for modelling need to be provided. Concepts such as recursion, structured programming, and even the correct use of the REPEAT statement do not occur automatically. Although a variety of bugs seemed to occur for all students, some specific difficulties are prominent in the work of individuals. An awareness of these difficulties and assistance in overcoming them could greatly speed that student's progress. Identifying and providing assistance to overcome persistent bugs could be undertaken jointly by the teacher and student.

The provision of manipulative aids and displays of Logo primitives, sample procedures, and editing commands is recommended as an integral part of the Logo environment.

If the teacher can help students reflect on and verbalize their computer experiences, growth and conceptualization could be facilitated. It appears from previous research and the results of this study that systematic planning ahead and looking back do not occur naturally for most students. It is also likely that transfer does not occur automatically, but must be encouraged by the teacher on behalf of the student. The opportunity for transfer of learning between pupil and pupil and between teacher and pupil could increase the knowledge of Logo and improve debugging strategies for both the transferer and the transferee.

REFERENCES

References

Alberta Education (1983). Elementary computer literacy. Edmonton, AB: Author.

Barton, J. M. (1984). Literacy is not enough: The computer as a bridge between psychological research and educational practice. AEDS Monitor, 23 (November/December), 17 - 18.

Breen, C. J. (1984). Microcomputers and mathematics: Can Papert help our schools? South African Journal of Education, 4 (3), 116 - 121.

Bruce, R. C. (1980). Software debugging for microcomputers. Reston, VA: Reston Publishing Company.

Cassel, D. & Swanson, R. (1980). BASIC made easy: A guide to programming microcomputers and minicomputers. Reston, VA: Reston Publishing Company.

Gates, R. (1983). Basic Debugging: A structured approach. Creative Computing, 9 (December), 169, 170, 173.

Gorman, H. Jr. (1983). Debugging Logo. 99'er Magazine, 2 (January), 42 - 44.

Groen, G. (1984). Theories of Logo. In R. J. Sorkin (Ed.), Logo 84. Cambridge, MA: Massachusetts Institute of Technology.

Hillel, J. (1985). Mathematical and programming concepts acquired by children, aged 8-9, in a restricted Logo environment. Interim research report. Montreal, PQ: Concordia University.

Kieren, T. E. (1984). Logo in Education: What, how, where, why and consequences. Edmonton, AB: Alberta Education.

Krasnor, L. R. & Mitterer, J. O. (1984). Logo and the development of general problem-solving skills. The Alberta Journal of Educational Research, 30 (June), 133 - 144.

Leron, U. (1985). Logo Today: Vision and Reality. The Computing Teacher, 12 (February), 26 - 32.

Martin, K., Bearden, D. & Muller, J. H. (1982). Turtle graphics on and off the computer. The Computing Teacher, 10 (November), 55 - 58.

Nelson, H. (1985). The pause that debugs. The National Logo Exchange, 3 (April), 1 - 2.

Nyberg, V. R. & Clarke, S. C. T. (1978). Technical report on the school subjects attitude scales. Edmonton, AB: Alberta Education.

Papert, S. (1980). Mindstorms: Children, computers, and powerful ideas. New York, NY: Basic Books, Inc.

Pea, R. D. (1984). Symbol systems and thinking skills: Logo in context. In R. J. Sorkin (Ed.), Logo 84. Cambridge, MA: Massachusetts Institute of Technology.

Perlman, R. (1976). Using computer technology to provide a creative learning environment for preschool children. Report No. LOGO-24. Washington, D.C.: National Science Foundation. (ERIC Document Reproduction Service No. Ed 207 576)

Shneiderman, B. (1980). Software psychology: Human factors in computer and information systems. Boston, MA: Little, Brown and Company.

Solomon, C. J. (1975). Leading a child to a computer culture. Artificial Intelligence Memo No. 343. Cambridge, MA: Massachusetts Institute of Technology.

Solomon, C. J. & Papert, S. (1976). A case study of a young child doing turtle graphics in Logo. Artificial Intelligence Memo No. 375. Cambridge, MA: Massachusetts Institute of Technology.

Sugarman, J. (1982). Brookline students hunt Logo bugs. The National Logo Exchange, 1 (December), 1 - 2.

Watt, D. (1979). Final report of the Brookline Logo project, part III: Profiles of individual student's work. Artificial Intelligence Memo No. 546. Cambridge, MA: Massachusetts Institute of Technology.

Watt, D. (1983). Learning with Logo. New York, NY: McGraw-Hill Book Company.

Weintraub, H. (1985). 100. The National Logo Exchange, 3 (April), 13.

APPENDIX 1

OBSERVATION SHEET

STUDENT _____ DATE _____ TIME _____

SETTING _____

PROJECT _____

_____ Uses modular prog.

_____ Test/debug each part

_____ Uses meaningful names

_____ Error Debugging

_____ Provoked Debug.

_____ Unprovoked Debug.

CLASSIFICATION	NOTES	STRATEGY

APPENDIX 2

Name _____

DEBUGGING COMPUTER PROGRAMS

1. worthwhile	___	___	___	___	___	worthless
2. not secure	___	___	___	___	___	secure
3. familiar	___	___	___	___	___	strange
4. happy	___	___	___	___	___	sad
5. deadly	___	___	___	___	___	lively
6. hard	___	___	___	___	___	easy
7. failure	___	___	___	___	___	success
8. student centered	___	___	___	___	___	teacher centered
9. like	___	___	___	___	___	dislike
10. practical	___	___	___	___	___	not practical
11. experiment	___	___	___	___	___	be told
12. useless	___	___	___	___	___	valuable
13. fair	___	___	___	___	___	unfair
14. fast	___	___	___	___	___	slow
15. unsure	___	___	___	___	___	sure
16. dull	___	___	___	___	___	interesting
17. unimportant	___	___	___	___	___	important
18. tense	___	___	___	___	___	relaxed
19. bad	___	___	___	___	___	good
20. active	___	___	___	___	___	inactive
21. enjoyable	___	___	___	___	___	dull
22. favorite	___	___	___	___	___	least favorite
23. unnecessary	___	___	___	___	___	necessary
24. abstract	___	___	___	___	___	concrete (real)

APPENDIX 3

NAME _____

STUDENT INTERVIEW

1. Completing a task (problem or procedure) that I start is
   a. very important
   b. important
   c. not important
   d. other

2. When I have difficulty doing a task I feel
   a. challenged
   b. unsure
   c. discouraged
   d. other

3. When I make errors I
   a. want to correct them immediately
   b. want to come back to them later
   c. never want to correct them
   d. other

4. When I have all the bugs out of my work I feel
   a. satisfied
   b. relieved but want easier work next time
   c. encouraged to try something harder
   d. other

5. When I have difficulty doing a task I
   a. quit
   b. try another approach
   c. look for materials to use
   d. ask a classmate for help
   e. ask a teacher for help
   f. other

6. How do you feel about bugs?

7. Do you feel you need or would like to know more about Logo?

8 What things would you like to know?

9. How can you find out more?

10. What is your most common kind of bug?

11. What is it that causes most of your bugs?

12. When you have bugs in your work what do you most often do to get rid of them?

___ Guess and check

___ Draw a diagram

___ Write notes

___ Calculate numbers

___ Ask teacher or friend

___ Look for a pattern

___ Modular programming

___ Consult a book or chart

___ Pencil turtle, turn chart

___ Use finger, hand, pencil

___ Play turtle, get up and walk

___ Follow line by line - notes

___ Follow line by line - graphic

___ Delete and start over

APPENDIX 4

University of Alberta
Edmonton, Alberta T6G 2G5
January 18, 1985

Dear Parent,

During the next several weeks I will be observing children at _____ School as part of a research project in Elementary Education, involving computer applications. I have chosen to tape record the children's comments as well as keep files on their computer work. Children will also be asked to respond to a short questionnaire regarding computer work.

This project has been approved by the University and Edmonton Public Schools. The signature of the principal of _____ School, affixed to this letter, will attest to the fact that I have his permission to conduct this research study.

If you have no objection to your child taking part in the study, would you kindly fill in the attached form and have it returned to your child's teacher.

Sincerely yours,

Mrs. Gloria Cathcart

_____
Principal,
_____ School

Please detach and return to your child's teacher.

I am willing to have my child, _____, take part in the research project in computer applications should he/she be selected to be a participant.

_____ _____
Date Parent's signature

APPENDIX 5

Complete Observation Schedule

Name	Obs	Date	Min	Project	Sup	Sub	Mng.N	Begin	Dev.Pro
Barry	1	Jan 21	30	BOX WITH TUN	1		No	Pro	Most
Barry	2	Jan 21	25	TUNNEL.CONT	1		No	Imm	None
Barry	3	Jan 28	30	SYMMETRY.3D.	1		No	Pro	Most
Barry	4	Jan 28	40	SYM.BOX.CONT	1		No	Pro	Most
Barry	5	Feb 4	50	SYM.CIR.FACE	1		No	Pro	All
Barry	6	Feb 4	25	SYM.FACE.EXP	2		No	Pro	All
Carol	1	Jan 21	30	3PRO	2		No	Pro	All
Carol	2	Jan 28	40	SYM.SET.SQ.	1		No	Pro	All
Carol	3	Jan 28	20	SYM.SQ.CONT	1		No	Pro	All
Carol	4	Feb 4	40	SHAPES	1		No	Exp	Most
Cindy	1	Jan 21	40	SUPER.3SUB	1	2	No	Pro	All
Cindy	2	Jan 28	30	S3S.FACE.CON	1	3	Some	Pro	All
Cindy	3	Jan 28	50	S3S.FACE.CON	1	5	Yes	Pro	All
Cindy	4	Feb 4	50	S3S.FACE.CON	1	8	Some	Pro	All
Connie	1	Jan 21	30	SUPER.3SUB		3	Some	Pro	All
Connie	2	Jan 28	25	S3S.CONT	1	3	Some	Pro	All
Connie	3	Feb 4	15	SYMMETRY		2	No	Pro	Half
David	1	Jan 21	30	THREE.PRO	3		No	Pro	All
David	2	Feb 4	25	MAKE.FACE	1		Yes	Pro	All
David	3	Feb 4	35	NEW.FACE	1		Yes	Pro	Most
Don	1	Jan 21	25	THREE.PRO	2		No	Pro	All
Don	2	Jan 21	30	THREE.PRO.NE	3		No	Pro	All
Don	3	Jan 28	10	SHAPES.SYM?	1		No	Pro	All
Don	4	Feb 4	20	SYMMETRY	1		No	Pro	All
Don	5	Feb 4	20	NEW.SYMMETRY	1		No	Pro	All
Doug	1	Jan 21	25	THREE.PRO	3		No	Pro	All
Doug	2	Jan 28	25	SYMM.BUTTERF	1		No	Pro	Most
Doug	3	Feb 4	25	SYMMETRY	1		No	Pro	All
Mark	1	Jan 24	20	HEART.ARROW	1	2	Yes	Pro	Most
Mark	2	Jan 31	30	ADD.LETTERS	1	5	Yes	Pro	All
Mark	3	Feb 1	25	ADD.LOVE	1	6	Yes	Pro	Most
Mark	4	Feb 7	30	SHAMROCK	1		Yes	Pro	Most
Mark	5	Feb 8	35	EASTER.EGG	1	1	Yes	Imm	Half

Name	Obs	Date	Min	Project	Sup	Sub	Mng.N	Begin	Dev.Pro
Mike	1	Jan 31	40	LOVE	1		Yes	Pro	All
Mike	2	Jan 21	35	ADD.HEART	1	1	Yes	Pro	All
Mike	3	Feb 1	25	HEART.ROTAT	1		Yes	Pro	All
Mike	4	Feb 7	20	HEART.R.CONT	1	1	Yes	Exp	Most
Mike	5	Feb 8	35	SYMMETRY	1		No	Pro	All
Murray	1	Jan 31	15	BE.MINE				Imm	None
Murray	2	Jan 31	25	BE.MINE	1		Yes	Pro	All
Murray	3	Feb 1	25	BE.M.CONT		2	Yes	Pro	All
Murray	4	Feb 7	30	BE.M.CONT	1	2	Yes	Pro	Most
Lana	1	Jan 24	20	VAL.CARD	1		Yes	Imm	None
Lana	2	Jan 31	30	CARD.CONT	1		Yes	Imm	None
Lana	3	Feb 7	30	CARD.CONT	1		Yes	Deb	None
Lana	4	Feb 8	25	CARD.CONT	1		Yes	Imm	None
Laura	1	Jan 24	30	LOVE	1		Yes	Imm	None
Laura	2	Jan 31	20	LOVE.CONT	1		Yes	Imm	None
Laura	3	Feb 7	40	CARD.SHAPE	1		Yes	Imm	None
Linda	1	Jan 31	45	PEACE	1	1	Yes	Imm	None
Linda	2	Feb 1	35	PEACE	1	4	Yes	Imm	Most
Linda	3	Feb 8	55	LOVE	1	4	Yes	Imm	None
Ray	1	Feb 11	15	CASTLE	1	2	Yes	Imm	Most
Ray	2	Feb 13	20	CASTLE.CONT	1	6	Yes	Pro	Most
Ray	3	Feb 14	20	CASTLE.CON	1	7	Yes	Pro	All
Ray	4	Feb 18	35	TESS.HEX	1	3	Yes	Pro	Half
Ray	5	Feb 25	45	TESS.TRI	1	3	Yes	Pro	Half
Sandy	1	Feb 11	25	CASTLE	1	2	Yes	Pro	Half
Sandy	2	Feb 11	35	CASTLE	1	5	Yes	Imm	Some
Sandy	3	Feb 18	30	TESS.I.HEX	1	3	No	Imm	Some
Sandy	4	Feb 25	40	TESS.HEX	1	5	Yes	Pro	Some
Shauna	1	Feb 13	55	SQ.IN.REC	1	7	Some	Pro	All
Shauna	2	Feb 18	20	TESS.I.HEX		2	Yes	Pro	All
Shauna	3	Feb 18	25	TESS.CONT	1	3	Some	Deb	Some
Shauna	4	Feb 19	50	TESS.CONT	1	7	Some	Deb	Some
Ted	1	Feb 13	30	FAN	1	1	Yes	Pro	All
Ted	2	Feb 14	20	FAN	1	1	Yes	Exp	All
Ted	3	Feb 18	30	TESS.I.HEX	1	1	Yes	Pro	Most
Ted	4	Feb 25	30	TESS.CONT	1	4	Yes	Imm	Most
Trevor	1	Feb 13	25	GRID	1	2	Yes	Pro	Half
Trevor	2	Feb 14	30	TUNL.FLOWER	2	2	Yes	Pro	Half
Trevor	3	Feb 18	50	TESS.I.HEX	1	3	Some	Imm	Some
Trevor	4	Feb 25	35	TESS.FRESH	1	3	No	Imm	Some

Name	Obs	Date	Min	Project	Sup	Sub	Mng.N	Begin	Dev.Pro
Troy	1	Feb 18	40	REC.SQUARE	1	3	Yes	Imm	Half
Troy	2	Feb 19	20	REC.S.CONT	1	3	Yes	Pro	All
Troy	3	Feb 25	35	REC#2	3	5	Yes	Imm	Most
Troy	4	Feb 25	15	REC.2.CONT	1	6	Yes	Deb	Most
Vicki	1	Feb 18	25	SQ.REC	3		Yes	Pro	All
Vicki	2	Feb 18	25	TESS.SQ	1	1	Yes	Pro	Half
Vicki	3	Feb 19	30	CHALL.3	2	2	Yes	Pro	Most
Vicki	4	Feb 25	30	SQ.IN.REC	1	2	Yes	Deb	All

Key to Headings

Obs - Observation number
Date - Date on which observation took place
Min - Length of observation in minutes
Project - Title or description of project
Sup - Number of superprocedures
Sub - Number of subprocedures
Mng.N - Uses meaningful procedure names
Begin - How session begins - procedural or immediate mode, debugging or experimenting
Dev.Pro - Amount of time spent developing in procedural mode

APPENDIX 6

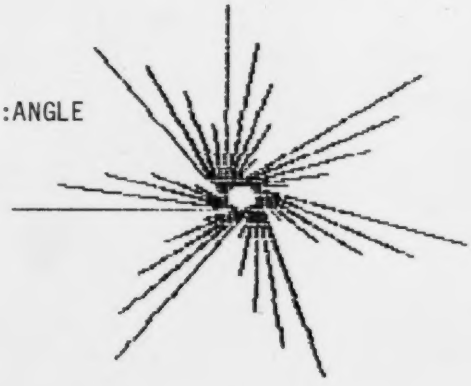
SCHOOL C ASSIGNMENTS

EXPLORING POWERFUL IDEAS

The FAN.LEFT procedure can be a very interesting procedure to explore the powerful ideas of recursion, variables and using conditional statements. This procedure can also be used as a building block for different procedures to create some beautiful designs.

1. What happens when you omit or change the condition
   IF :DISTANCE < 0 [STOP] ?
2. What happens when you change the variable that controls the angle?
3. What happens when you try different inputs for distance and angle?
4. Create other procedures that can use the FAN.LEFT procedures.

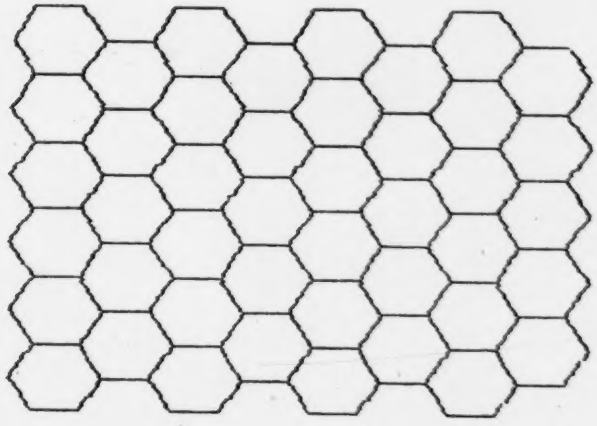
```
TO FAN.LEFT :DISTANCE :ANGLE
IF :DISTANCE < 0 [STOP]
FD :DISTANCE
BK :DISTANCE - 5
LT : ANGLE
FAN.LEFT :DISTANCE - 3 :ANGLE
END
```



EXPLORING TESSELLATIONS

CHALLENGE!!!!!

Construct procedures to make a design similar to the design on this page. Remember, it is often wise to break a large problem down into simpler, more easily solved problems.



What other shapes can you tessellate? Try combining shapes and tessellate them.

LOGO CHALLENGE

Define a procedure that will draw rectangles with different widths and lengths.

Define another procedure that will draw different sized squares.

Problem #1

Using procedures, construct a rectangle 160 steps be 200 steps. Cover this rectangle with squares 40 steps by 40 steps. How many 40 by 40 squares will cover this rectangle?

Problem #2

Construct another rectangle 200 by 250 steps. Cover this rectangle with squares 25 steps by 25 steps.

Problem #3

Construct procedures that will position the turtle. Try to use the Logo command REPEAT. Use variables. Make your variable names descriptive if possible.

APPENDIX 7

SAMPLE COMPLETED OBSERVATION SHEETS AND LOG NOTES

STUDENT Mike DATE February 1 TIME 11:20 to 11:45

SETTING In unoccupied classroom adjacent to regular classroom

PROJECT Making a small heart, repeating the heart for a design

No	Uses modular prog.	Yes	Error Debugging
Yes	Test/debug each part	No	Provoked Debug.
?	Uses meaningful names	Yes	Unprovoked Debug.

CLASSIFICATION	NOTES	STRATEGY
	Mike begins in procedural mode. TO H HT PC 3 REPEAT 228 [RT .5 FD 1] END Tests	
	"I think it might be big. I know how to make it small." Edits - TO H PC 3	
1. *Dimension - arc size*	REPEAT 228 [RT 6 FD 1] END Tests	*Guess and Check*
	"Too small. Nice circle." Edits - TO H PC 3	
Dimension - No. of Repeats arc size	REPEAT 30 [RT 4 FD 1] PC 0 HOME PC 3	*Guess and Check*
	REPEAT 30 [LT 4 FD 1]	
2. *Length*	FD 20 -- Changes to FD 40 END Tests	*Guess and Check*
	"Not too bad so far." Edits, changes PCs to PU and PD, and changes two lines.	
Dimension - arc size	REPEAT 30 [RT 2 FD 1] PU HOME PD	*Guess and Check*
	REPEAT 30 [LT 2 FD 1] Tests	

	"I was thinking of making a circle of hearts. I have to make the REPEAT go more." Edits the same two lines.		
Dimension - No. of Repeats	REPEAT 100 [RT 2 FD 1]		
	REPEAT 100 [RT 2 FD 1]	Tests	*Guess and Check*
	"Almost!" Edits the same two lines.		
Dimension - No. of Repeats (5 tries)	REPEAT 115 [RT 2 FD 1]		
	REPEAT 115 [RT 2 FD 1]	Tests	*Guess and Check*
	"Make it FORWARD 30." Edits to change last line.		
Length	"No. It was 40 before. Make it FORWARD 50."		*Guess and Check*
	FD 50		
	RT 77.5		
	FD 50	Tests	
	"Oh, wrong way. It's suppose to be LEFT." (Checks procedure) "Yah, it's suppose to be LEFT." Edits second last line.		*Observe and Change*
	LT 77.5	Tests	
3. Lateralization			
	"I don't think I got it far enough. No, it should go FORWARD 55." Edits		
Length	FD 55		*Guess and Check*
	LT 77.5		
	FD 55	Tests	
	"Close enough. No, just a little more." Edits		
Length (4 tries)	FD 60		*Guess and Check*
	LT 77.5		
	FD 60	Tests	
	"Oh, almost!" Edits		
	FD 60		
4. Orientation	LT 78.5		*Guess and Check*
	FD 60	Tests	
	"Oh! Couldn't believe it!"		
	(Mike's procedure was saved to disk at this point and appears following this observation sheet under MIKE1)		

Goes into editor and adds
REPEAT 5 [H] Tests

"See it goes to there and then goes to HOME. I think I can fix it."
Edits

```
TO H
HT
PC 3
REPEAT 115 [RT 2 FD 1]
PU
HOME
PD
REPEAT 115 [LT 2 FD 1]
FD 60
LT 78.5
FD 60
REPEAT 5 [H]
END
```

5. *HOME does not allow rotation*

Mike changes the sequence of commands to avoid using HOME. He deletes PU HOME PD, inserts the three lines, changing the LT to RT, changes the LT in the REPEAT line to RT, and then deletes the three lines preceeding the last REPEAT. He does not hesitate throughout this whole process. *Aha!*

(Procedure is saved on disk at this point and called MIKE2.)

```
TO H   (Edited procedure.)
HT
PC 3
REPEAT 115 [RT 2 FD 1]
FD 60
RT 78.5
FD 60
REPEAT 115 [RT 2 FD 1]
REPEAT 5 [H]
END
```

Tests

Mike is very pleased as he sits and watches his procedure work. Finally he realizes that it is not going to stop. He seems to understand what is happening (recursion) but not what to do about it. He edits again and changes the last line.

Unprovoked debugging -exploring

REPEAT 5 [RT 35 H] Tests

"That's what I call pretty weird!"
Session ends.

(Procedure from the disk is found under MIKE3.)

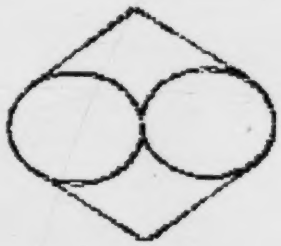
Mike's Procedures and Graphics From Disk

MIKE1

```
TO H
HT
PC 3
REPEAT 115 [RT 2 FD 1]
PU
HOME
PD
REPEAT 115 [LT 2 FD 1]
FD 60
LT 78.5
FD 60
END
```

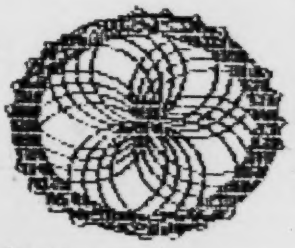
MIKE2

```
TO H
HT
PC 3
REPEAT 115 [RT 2 FD 1]
FD 60
RT 78.5
FD 60
REPEAT 115 [RT 2 FD 1]
REPEAT 5 [H]
END
```



MIKE3

```
TO H
HT
PC 3
REPEAT 115 [RT 2 FD 1]
FD 60
RT 78.5
FD 60
REPEAT 115 [RT 2 FD 1]
REPEAT 5 [RT 35 H]
END
```



OBSERVATION SHEET

STUDENT Sandy DATE February 11 TIME 2:40 - 3:05

SETTING In computer lab, at one end. Another class in the lab

PROJECT Designing a castle

Yes	Uses modular prog.	Yes	Error Debugging
Yes	Test/debug each part	____	Provoked Debug.
Yes	Uses meaningful names	____	Unprovoked Debug.

CLASSIFICATION	NOTES	STRATEGY
	Enters in teach mode: TO SETUP >SETPOS [0 -80] >END — Didn't test.	
1. *Omission* *PU - PD*	Edits TO SETUP PU SETPOS [0 -80] PD END — Tests.	*Observe Listing and Change*
2. *Length*	In immediate mode: FD 50 FD 50 RT 90 FD 10 RT 90 FD 10 LT 90 FD 10 RT 90	*Immediate Mode* *Guess and Check*
3. *Lateraliz-ation*	LT 180 FD 10 RT 90 FD 10 RT 90 FD 10 LT 90 FD 10 LT 90	*Immediate Mode* *Observe and Change*

"I know I've got to keep doing this so . . ." She goes to the editor.

TO WALL — *Sees Pattern*
FD 100
RT 90
FD 10
RT 90
FD 10
LT 90
FD 10
LT 90
FD 10
END
Sandy sees that she could have used repeat so goes in after FD 100 and adds:

4. *Semantic - REPEAT*

REPEAT 10 [] — She tests. — *Guess and Check*

Error Message

Error message. Sandy understands, realizes her mistake, and using the delete key, removes the last bracket from the REPEAT. She then deletes the RETURNs between each line, drawing them up into the REPEAT statement.

TO WALL
FD 100
REPEAT 10 [RT 90 FD 10 RT 90 FD 10
LT 90 FD 10 LT 90 FD 10] Tests.

"Too many times." She counts the parts before it wraps.

5. *Dimension - REPEAT*

Edit. Changes REPEAT 10 to 7 — *Guess and Check*
Tests SETUP WALL

Dimension

Edit. Changes REPEAT 7 to 6 — *Guess and Check*
Tests

In immediate mode: — *Immediate Mode*
RT 90
FD 10
RT 90

6. *Length*

FD 100 — *Guess and Check*
RT 90
FD 100
FD 20

(This finishes off the top, makes the right side and the bottom.)

Edit. Sandy thought one more REPEAT should fit on the screen so she tries again.

Dimension Changes REPEAT 6 to 7 and tests. *Guess and Check*

Dimension Changes REPEAT 7 back to 6. *Guess and Check*

Writes superprocedure.

```
TO S
SETUP
WALL
END
```

In immediate mode again tries: *Immediate Mode*

```
RT 90  FD 10
RT 90 FD 100
RT 90 FD 140
```

Edits, adding the above into the end of the REPEAT statement. Should go after the REPEAT.

```
TO WALL
FD 100
REPEAT 6 [RT 90 FD 10 RT 90 FD 10 LT
90 FD 10 LT 90 FD 10 RT 90 FD 10 RT
90 FD 100 RT 90 FD 140]
END
```

7. *Logic (Miscell.)* Tests.

Studies listing. "No wonder!" Sees that commands should come after the REPEAT but does not take it all back out. (The listing of Sandy's work to date, taken from the disk appears on the page following the observation under SANDY1) Her WALL procedure now reads: *Listing* *Aha!* *Guess and Check*

Removes some

```
TO WALL
FD 100
REPEAT 6 [RT 90 FD 10 RT 90 FD 10 LT
90 FD 10 LT 90 FD 10 RT 90 FD 10 RT
90]
FD 100
RT 90
FD 140
END
```

Sandy tests and at this point takes a pencil and graph paper to help figure out her bug. She follows the listing, drawing as she goes until she comes to the end of one REPEAT. She then reinserts the FD 100 and tests again. She sees where the REPEAT should end, deletes, and tests. She now adds some of the commands after the REPEAT as they should be.

Pencil and graph paper

Puts some back

Guess and Check

Removes all excess

Guess and Check

```
TO WALL
FD 100
REPEAT 6 [RT 90 FD 10 RT 90 FD 10 LT
90 FD 10 LT 90 FD 10]
```

Adds after

```
RT 90
FD 10
RT 90
FD 100
END                          Tests.
```

Sandy counts the 10's.
Edits and adds before the END -

```
RT 90
FD 130
HT
END                          Tests S.
```

Sandys's procedures are again saved on the disk and are as listed on the following page under SANDY2.

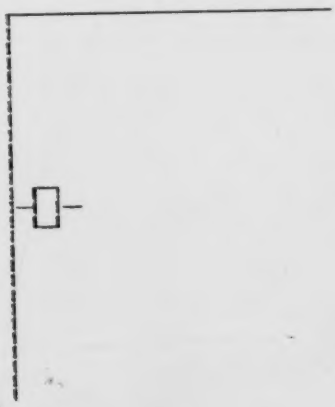
Sandy's Procedures and Graphics From Disk

SANDY1

```
TO WALL
FD 100
REPEAT 6 [RT 90 FD 10 RT 90 FD 10 LT 90 FD 10 LT 90 FD 10 RT
90 FD 10 RT 90]
FD 100
RT 90
FD 140
END

TO SETUP
PU
SETPOS [0 -80]
PD
END

TO S
SETUP
WALL
END
```

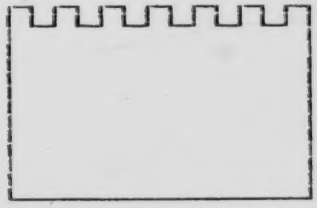


SANDY2

```
TO WALL
FD 100
REPEAT 6 [RT 90 FD 10 RT 90 FD 10 LT 90 FD 10 LT 90 FD 10]
RT 90
FD 10
RT 90
FD 100
RT 90
FD 130
HT
END

TO SETUP
PU
SETPOS [0 -80]
PD
END

TO S
SETUP
WALL
END
```



Log Notes on Sandy

February 11

Sandy is a very confident appearing little girl, who says, "I know." She started right in in teach mode, wrote a setup procedure and then before testing went back into edit mode to add PU and PD which she had omitted.

She handles the editor with ease -- moves a number of single command lines into one line in the REPEAT statement.

Worked all through this first part without making an error except for a RT 90 in immediate mode which was quickly changed by LT 180, and didn't reoccur in edit mode. Sandy used REPEAT 10 which wrapped around and then guess and test to find right size. Sandy took no notes, but looked at immediate mode steps -- went to the editor and entered the steps from memory.

February 14

Sandy tried several things in immediate mode but did not follow through with most of them. The TOP she did try was not balanced so she was not pleased with it. We left it with the top askew.

Sandy is a bubbly girl with lots of ideas. Her teacher says she is very popular. She was excited today -- Valentine's and a field trip.

February 18

Sandy came smiling, eager to try out the new task assigned by her teacher. She started in immediate mode after reading the assignment. She followed the diagram closely, starting with a RT 45 and putting two RT 45's together in the center.

She began typing the information into a procedure and after the third line hit <CONTROL> G, and with a fresh start used REPEAT 6 [RT 45 FD :SIZE] END and then into the editor to add the variable to the name. She had to this point worked in the teach mode.

Sandy strikes me as a bright, fairly impulsive girl. She is very bubbly and her thoughts seem to race ahead. Taking time to stop and think things through more thouroughly today may have helped her. When the 30 minutes were up Sandy had two procedures that linked two shapes together, each in a different way, but no succes with a row.

February 25

Sandy came to work at 10:00 when the others went to the gym to organize for the sports, as she was not going to the gym. She chose to stay and work through recess as well. This whole session was taped. She chats away quite freely. She knew this time that she wanted a regular 6 sided shape, but did not know the turn required, or how to find it, except by experimenting, which she did in a fairly organized fashion. Since her shape started with a FD rather than a turn, she seemed to handle it better.

Although Sandy was interested in writing a row procedure, her efforts were quite circular. She was very aware of the tremendous amount of overlapping that was taking place but did not solve this problem in the time that she worked (40 min.). She did get some interesting designs, which she enjoyed, even though she did not let them take her off task.

APPENDIX 8

Bug Specimen

Samples of student work illustrating the various bugs will be included and discussed in this section. An attempt has been made to choose bugs which are representitive of those found in the work of others as well as the student from which it was taken. In some cases asterisks are used to mark the bugs being discussed.

Dimension Bugs

The work of Barry, as he designed a pair of three dimensional boxes (later changed into a face) as an example of symmetry, is a typical example of the occurrance of dimension bugs. Barry worked for one 25 minute period during which five dimension errors occurred. Although the procedure contained two REPEAT commands they were free of errors and all the bugs were in determining lengths for other than the front face of the cubes.

```
TO JOE
REPEAT 4 [FD 90 RT 90]
END
```

Barry tested his procedure and then worked in immediate mode.

```
 FD 90
 RT 45
*FD 20
*FD 20
 LT 45
 RT 90
*FD 40
*FD 20
*FD 20
 LT 45
 RT 45
 RT 45
 RT 45
```

"I'll go back to edit," he said.

```
TO JOE
REPEAT 4 [FD 90 RT 90]
FD 90
RT 45
*FD 20
```

Barry did not remember the immediate mode work. Notes at this point would have helped immensely. He tested and then reentered the editor, changed the FD 20 to FD 40 and added:

```
RT 45
*FD 50
RT 90
```

He then changed FD 50 to FD 70 to FD 90

Cindy's 40 minute period in which she worked with three circles to design a head was representative of the problems in working with circles in Terrapin Logo. Her large circle was made by REPEAT 90 [FD 30 RT 30] while the small one was REPEAT 30 [FD 10 RT 30]. In an attempt to make the small circle smaller the following lines were entered and tested;

```
REPEAT 90 [FD 10 RT 10]
REPEAT 90 [FD 10 RT 30]
REPEAT 30 [FD  5 RT 30]
```

The desired result was achieved but no heed was paid to the number of times the circle was retraced or the stopping position of the turtle. Most students observed using circles appeared not to have a good grasp of how to make the circle the size they desired or what parts of the REPEAT command to adjust to get the desired results.

Trevor's 25 minute period spent designing a grid made use of three REPEAT statements and illustrated a clear understanding of their function. His only bug was in determining how many times he needed to repeat the particular part of his very compact modular program.

```
TO SQUARE  :SIZE
REPEAT 4 [FD :SIZE RT 90]
END
```

He tested twice by using SQUARE 4, CS, SQUARE 10. He then worked in immediate mode.

```
FD 10
SQUARE 10
FD 10
SQUARE 10
*REPEAT 10 [FD 10 SQUARE 10]
*REPEAT 4 [FD 10 SQUARE 10]
*REPEAT 6 [FD 10 SQUARE 10]
```

Trevor continued, determining how to start the next row of squares. Following his immediate mode work Trevor knew how many REPEATs he needed, but it did take time to solve this dimension bug. He then entered the following subprocedure:

```
TO LINE
RT 90
FD 10
LT 90
REPEAT 24 [FD 10 SQUARE 10]
END
```

He then faced the difficulty of determining the number of lines needed for his graph, and proceded to handle that problem in the same manner.

```
LINE
REPEAT 4 [LINE]
REPEAT 24 [LINE]
```

"Too many times." The design wrapped around the screen and over itself. Trevor counted the squares twice, but found it difficult to keep track.

```
TO GRAPH
REPEAT 23 [LINE]
HOME HT
END
```

He tested and then edited it to:

```
REPEAT 27 [LINE]
```

"I had to guess approximately how many squares it would make." He tested and then edited again.

```
REPEAT 28 [LINE]
```

Thus his final bug free procedure was:

```
TO GRAPH
REPEAT 28 [LINE]
HOME HT
END
```

Although Trevor was considered a knowledgable, very efficient programmer, he faced dimension bugs. It was not surprising then, that dimension bugs were prevalent problems for most students.

Orientation Bugs

A number of orientation bugs occurred for Shauna when she was trying to tessellate an irregular hexagon. Even the designing of the hexagon itself presented orientation bugs. After removing the lateralization bugs Shauna had:

```
TO SHAPE
LT 45
FD 20
LT 45
FD 20
LT 45
FD 20
LT 45
FD 20
LT 45
FD 20
```

She tested this and said, "Oh, boy! Does it look the same?" "I got it!" She edited, changing the fourth LT 45 to LT 90 and tested again. Then it was necessary to add LT 90 FD 20. The orientation bugs prevailed throughout the establishment of a tessellation, as

illustrated by the following short segments. She had two subprocedures, SHAPE and ALL and was trying to use them repeatedly.

```
SHAPE ALL       SHAPE ALL       SHAPE ALL
RT 90           RT 90           RT 90
FD 20           FD 20           FD 20
RT 45           RT 45           RT 45
FD 20           FD 20           FD 20
LT 45           ALL             SHAPE
SHAPE
```

In each of these attempts the orientation was not correct.

Mike's orientation bug cited here occurred when he was trying to make the letter V. When at the bottom of the first stroke he entered LT 45. This was changed to LT 140 and then to LT 175. LT 140 was already too large so LT 175 was a very inappropriate attempt. Mike then entered LT 100 and commented, "Getting there. That looks pretty good." He then entered LT 105 and then LT 103.5, and said, "Looks pretty good."

Lateralization Bugs

Many of the lateralization bugs occurred on 90 degree turns. The following short excerpt from the immediate mode work of Lana contains two lateralization bugs and her correction of each.

```
FD 70
PD
LT 90
FD 20
RT 90
FD 10
RT 90
FD 10
BK 10
*RT 90
*LT 180
 FD 10
*LT 90
*RT 180
 FD 20
```

When working with the tessellation problem, Shauna ended a section with RT 45 and then wished to call a subprocedure. When it was tested, Shauna pointed and said in disgust, "Right is that way. Left is that way." She then changed the RT 45 to LT 45.

Origin Bugs

The examples included in this section are drawn from the work of 4 students who encountered bugs in determining the origin for the procedure or superprocedure.

Vicki faced difficulties setting the origin for her task. First she did not see that the reason her large rectangle did not fit the screen was due to the origin point being at HOME. She said of her teacher, "Why does he want it so big! The screen isn't that big." When she realized that a new origin point must be used, she edited her rectangle procedure, entering at the beginning:

```
PU
LT 90
FD 60
```

When she tested she said, "Oh, I forgot PENDOWN. I think it will work." She added PD and tested again. She responded, "Oh, now I have to go here," and pointed to the bottom left corner of the screen. Following the FD 60 she added:

```
 LT 90
 FD 35
*LT 90
 PD
```

Instead of changing the LT 90 to LT 180 or RT 180, the more common approach (turn-move-turnback), Vicki changed the first turn at the beginning of the rectangle construction.

Linda's project was to write LOVE, starting near the upper left corner of the screen. She set the pen color to one and began:

```
LT 90
RT 30
FD 50
```

At this point she typed DRAW and started again.

```
PU
LT 65
FD 100
FD 50
PD
FD 20
```

Again she typed DRAW and started over.

```
PU
RT 90
LT 180
FD 100
FD 30
RT 90
FD 60
PD
FD 20
```

This time she was pleased and so entered the necessary steps into the editor in order to establish the origin, and then added the part to make the letter L.

```
TO L
PU
LT 90
FD 130
RT 90
FD 60
PD
FD . . .
```

When looking at the immediate mode work of Linda, one can see how the origin bug is linked with orientation and dimension bugs. Her work also showed her problem with lateralization.

While designing an egg, Mark began his procedure:

```
TO EGG
PU
SETY 100
. . .
```

This was then edited to SETY 90 and then to 80. Later it was edited to SETY (-80) when Mark decided the egg would be better near the bottom of the screen.

Ted used a SETUP procedure with variable inputs to determine the origin for tessellating his irregular hexagon. The procedure was first written with one variable, used to determine the number of times the shape, called PENT, would fit across the screen. Upon using it, Ted realized that he didn't know how much to go forward between each shape in order to establish the origin location for the second shape. Therefore he added a second variable and tested.

```
TO SETUP :P
REPEAT :P [LT 90 PU FD 45 RT 90 PD PENT]
END
TO SETUP :P :C
REPEAT :P [LT 90 PU FD :C RT 90 PD PENT]
END
```

He then tested a number of values.

```
SETUP 15 25     CS
SETUP 15 30     CS
SETUP 15 27     CS
SETUP 15 33     CS
SETUP 15 34     CS
```

"Almost."

```
SETUP 15 35     CS
```

"That's it, I think." At this point Ted was letting the shapes wrap around the screen and was trying to get them to overlap perfectly. He later established that seven repeats were needed to fill the screen,

going across. The variables were moved from the procedure and it was written:

```
TO SETUP
REPEAT 7 [LT 90 PU FD 36 RT 90 PD PENT]
END
```

Ted then worked in order to establish the origin point for the whole process to begin. Once he solved that bug, he decided on procedure names and first wrote the superprocedure.

```
TO PENTS
SETUP1
SETUP
SETUP2
```

"I think that's all I need."

```
END
```

He then wrote SETUP1 as follows:

```
TO SETUP1
PU SETPOS [120 0]
END
```

He tested PENTS. All was fine except that he received the message, "I don't know how to SETUP2." Some more immediate mode work followed and then the procedure, called SETUP2 was written in order to make the transition between the horizontal row of shapes and the vertical row.

```
TO SETUP2
PU
FD 36 LT 90
END
```

There were still bugs to be worked out in finding the origin. The location in SETUP1, although tested in immediate mode, was not satisfactory. Ted followed his previous style, edited the procedure, added a variable and tested.

```
TO SETUP1 :P
PU SETPOS [120 :P]
END
```

Upon testing, the message "SETPOS DOESN'T LIKE :P AS INPUT" resulted. This was understood by Ted and the offending part removed. He then edited and tested until he was satisfied. His next step, although it does not relate to an origin bug, is noted here. Ted saw a need to have a variable back in SETUP so that it can be used to make both the vertical row of shapes as well as the horizontal one. He edited both SETUP and PENTS. At the close of the session the procedures were saved and the listing and graphic appear in Figure 10.

Figure 10

Ted's Tessellation

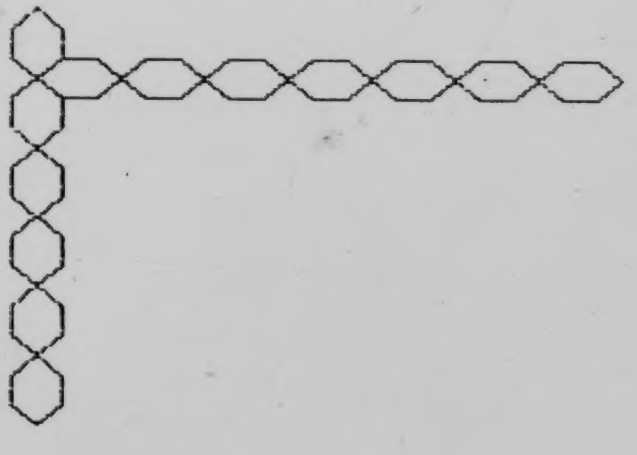
```
TO PENT
RT 45 FD 15 RT 45 FD 15 RT 45 FD 15 RT 90 FD 15 RT 45 FD 15
   RT 45 FD 15 RT 45
END

TO SETUP :P
REPEAT :P [LT 90 PU FD 36 RT 90 PD PENT]
END

TO SETUP1
PU SETPOS [125 80]
END

TO SETUP2
PU
FD 36 LT 90
END

TO PENTS
SETUP1
SETUP  7
SETUP2
SETUP  6
END
```



Omission Bugs

Although the omission bugs were minor in nature and usually easily corrected, they did occur in a number of settings. The PENUP and PENDOWN omissions were evident in the move procedure, whether the move was executed with SETPOS, SETX and SETY, or a turn-forward-turnback approach. The samples cited here are typical of those occurring.

Ray's omission bug arose in the following procedure.

```
TO MOVE
SETPOS [-100 -100]
END
```

After testing, Ray exclaimed, "Oh, I forgot PENDOWN!" The procedure was edited to read:

```
TO MOVE
PU SETPOS [-100 -100] PD
END
```

Ray was a very efficient programmer when compared to his classmates, but was not exempt from even the simplest of bugs. Bugs of this nature were easily dispensed with and caused him little concern.

When positioning a subprocedure, Mark entered the following:

```
TO V
PU
HOME
SETX 0
SETY (-100)
RT 135
FD 18
LT 90
FD 18
```

He tested and said, "I forgot PENDOWN." Mark edited and added PD after SETY (-100). Remembering to include PU does not insure the inclusion of PD. This work also contained an example of the use of

excess information which was not counted as a bug since it did not interfere with the design. It was not necessary for Mark to use both the HOME and SETX 0 commands. One or the other would have been sufficient.

Troy's work also has a PENDOWN omission.

```
TO MOVE2
RT 90 FD 25 LT 90 PU BK 150
END
```

After testing Troy said, "Oh, I forgot PENDOWN." Troy edited and added PD to the end of the line. MOVE, written the day before had the same bug in it. It is interesting to note the comment uttered by each of these students upon testing their procedure. It seems that the bug is really one related to forgetting and not to misunderstanding.

Vicki omitted the FORWARD command in her tessellation of squares (which she misspelled). While concentrating on the number of REPEATs needed, she had commented, "I don't know how many it will take to fill the screen."

```
TO SQ
REPEAT 16 [RT 90 SQARES]
END
```

After testing Vicki said, "You have to move it over cause it will just keep going RT 90, RT 90." Vicki edited, changing the REPEAT and adding a FD 45.

```
TO SQ
REPEAT 4 [RT 90 SQARES]
FD 45
REPEAT 4 [RT 90 SQUARES]
END
```

Murray had entered the procedure MINE from his notes. Upon testing he commented, "Oh, wait a minute. I think I forgot to write

something down." He studied the listing and his notes, and in fact, he had omitted RT 135 PU FD 33. This type of bug occurred in the work of all the students who were consistent note takers.

Excess Information Bugs

Many of the instances of excess information did not adversely affect the design so are not considered as bugs. PENUP and PENDOWN errors were usually omission bugs, but also appeared as excess. The work of Barry is an example of this. Only a part of his long procedure is listed here.

```
 TO PRO
*REPEAT 3 [FD 90 LT 90]
*PU
*HOME
*PD
 REPEAT 4 [FD 90 LT 90]
*PU
*HOME
*PD
 REPEAT 4 [FD 25 LT 45]
 FD 25
**PU
 LT 90
 FD 50
 LT 45
*PD
 FD 50
 LT 45
 FD 90
```

When the procedure was tested, Barry saw that the PU** was excess and so he removed it. Notice all the other excess commands, marked with a single asterisk, in this section.

Troy's work with rectangles illustrated his problem with excess information relating to variable inputs.

```
TO REC2  :SIZE  :WIDTH
FD :SIZE
RT 90
FD :WIDTH
RT 90
FD :SIZE
RT 90
FD :WIDTH
END
TO REC3
PU
BK 110
LT 90
FD 50
PD
RT 90
*REC2 200 160 200 160
HT
END
```

When he tested REC3, Troy received the message, "I don't know what to do with 200 in REC3." Neither did he!

After making a rectangle Shauna made a square as follows:

```
TO SQ :W :L
REPEAT 2 [FD :W RT 90 FD :L RT 90]
END
```

After testing it she realized that she needed only one variable input to make a square so she edited the procedure.

```
TO SQ :SIZE
REPEAT 2 [FD :W RT 90 FD :L RT 90]
END
```

This, of course, did not work so she returned it to the original form and moved on to the next part of her assignment. This example and the one before it not only illustrate excess information, but a lack of understanding of variable inputs.

Syntax Errors

The bugs relating to syntax errors were not numerous, especially when one considers that none of the students observed had received

formal keyboarding instruction, but used the "hunt and peck" method of typing. These bugs were generally easily identified by the students and easily rectified. The following are taken from the work of a number of different students. The spelling of REPEAT (typed REPAET) was the bug of 2 students. CUPID was mistyped as CUPIT. Other typing errors included RT 9 for RT 90, FD 5 for FD 50, FD 50 for FD 5, and ENDC for end. The omission of a space resulted in the following: 25LT, 180FD, 20END, 1FD, :SIZERT. Two occurances of unmatched brackets and one misplaced bracket gave error messages. The incorrect syntax for SETXY also occurred. The attempt to use 17 as the name of a procedure and :P as an input value for SETPOS were also observed as bugs. The shifted brackets in Terrapin Logo represented the most difficult bug to remove since the two brackets visually appear the same but are understood differently by the computer.

Miscellaneous Bugs

A number of students made use of HOME, SETPOS, SETX and SETY or their own MOVE definition within their procedures or subprocedures. On many occasions this did not present a problem, but for 5 students, when they tried to use their subprocedures again on a different part of the screen, bugs occurred.

Mike's work illustrated the offending HOME command. He was unable to repeat the heart in a new location.

```
TO H
HT
PC 3
REPEAT 115 [RT 2 FD 1]
PU
HOME
```

```
PD
REPEAT 115 [LT 2 FD 1]
FD 60
LT 78.5
FD 60
END
```

Mike was willing to debug his procedure to get the desired results.

Mark, when faced with a similar bug abandoned the idea and changed the project. His procedures follow.

```
TO EGG
PU
SETY (-80)
PD
RT 180
REPEAT 22.5 [FD 5 RT 8]
FD 10
REPEAT 11 [FD 5 RT 5]
PU
HOME
SETY (-80)
PD
REPEAT 12 [FD 7 LT 5]
SETY (-10)
END
```

```
TO EGGS
SETY (-10)
LT 180
EGG
END
```

He tested EGG, then EGGS. Ray's bug occurred when he embedded his MOVE procedure in a procedure called P. He then tried to repeat the procedure P and the bug surfaced. He was able to remove the MOVE procedure from P and then use both procedures appropriately in a superprocecure.

Using the words TO and ED within a procedure are not acceptable. Cindy and Barry respectively discovered this. Cindy was calling a subprocedure and instead of typing the name JA she typed TO JA. Barry entered ED within his procedure as he thought that it would make the whole screen flash off and then back to his graphic.

Some semantic problems with REPEAT occurred. These were very evident in the work of Doug.

```
TO BUTTERFLY
REPEAT 2 [ RT 45 FD 20 RT 80 FD 10 RT 35 FD 30 RT 35 FD 40]
[RT 45 FD 10 RT 45 FD 20 RT 45 FD 10]
REPEAT 2 [ LT 45 FD 20 LT 80 FD 10 LT 35 FD 30 LT 35  FD 40]
END
```

He received the error message, "You don't say what to do with [RT 45 ..." - his middle line. He did not understand this, nor did he understand the contents of the two long REPEAT statements. Doug also attempted to use commands such as PU 50 and PR 60 to get the turtle to go 50 steps toward the top of the screen and 60 steps to the right of the screen.

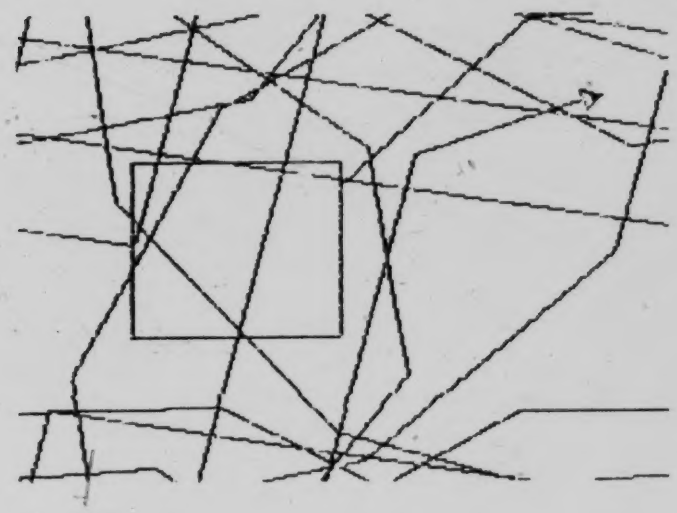
In observing Don's use of REPEAT statements in was not clear whether he fully understood the semantics involved, although he did on some occasions use them correctly. One of his procedures and the graphic appear in Figure 11. Don was likely "performing" at this stage and certainly did not have a clearly planned project in mind. It is interesting to note that his double digit input values are all adjacent keystrokes.

The LOGO SYSTEM BUG which occurred in Troy's work on came as a mystery and was totally crippling. The language had to be reloaded.

Figure 11

Don's Graphic

```
TO D.L.
REPEAT 4 [FD 45 LT 90 FD 45]
REPEAT 7 [FD 32 RT 43 FD 87]
FD 65
REPEAT 9 [FD 54 LT 32 FD 98]
FD 65
RT 87
FD 90
FD 800
LT 87
FD 540
RT 56
FD 76
END
```



APPENDIX 9

Debugging Strategy Specimen

A representative sample of student work illustrating the various debugging strategies will be included and discussed in this section. In most cases the strategy is discussed in relation to the particular bug the student was trying to eliminate.

Immediate Mode

The work of Linda is representative of the work of 3 students, all girls and all from School B, who first worked in immediate mode, taking notes as they went along. Then the work was transfered from the notes into the editor, tested and if necessary, further debugged. Linda's work goes one step beyond that of the other two, since she not only planned her work in modules but made use of subprocedures.

The following is taken from the work of Linda, as she started her Valentine's Day project in which she designed the large letters L O U E to go diagonally across the screen.

Types	Writes
	L
PU	PU
RT 90	
LT 180	LT 90
FD 100	
FD 30	FD 130
RT 90	RT 90
FD 60	FD 60
PD	PD
FD 20	FD 20
BK 20	BK 20
LT 90 (erases by backspacing)	
RT 90	RT 90
FD 10	

In this sample of work, Linda's notes accurately represent what she wanted in her procedure. This is not always the case as is evidenced in Laura's work in the section under multiple strategies near the end

of this appendix. Students taking notes from their immediate mode work to transfer into the editor do face difficulty. The omission of lines or incorrect copying is not uncommon. During an observation period with Laura, she used <CONTROL> T to get the text screen and check her work, but missed the top line which had scrolled off. Lana, who also did a considerable amount of immediate mode work, often typed with her left hand while writing with her right. She also devised a system of entering four lines, copying four lines, and then filling the four lines with question marks by hitting the RETURN key four times. When the last question mark disappeared as the next four lines were typed, it was again time to copy the lines into her notebook.

In the work of some students, such as Troy's work, a fairly large block of time was spent in immediate mode, with no notes being taken by the student. The student then attempts to transfer the results to procedural mode, and often faces some of the same difficulties that were just resolved in immediate mode. The experience of having worked in immediate mode seemes beneficial in that problems were resolved much more quickly the next time. On two occasions Barry asked the researcher to take notes for him, which was done. Murray's comment, while working in procedural mode was, "I don't really know what I'm doing. I should have written it down."

Some students have very good recall of immediate mode work, as evidenced by Sandy's work which is displayed below. In immediate mode she typed:

```
FD 50
FD 50
RT 90
FD 10
RT 90
FD 10
LT 90
FD 10
RT 90
LT 180
FD 10
RT 90
FD 10
RT 90
FD 10
LT 90
FD 10
LT 90
```

```
TO WALL
FD 100
RT 90
FD 10
RT 90
FD 10
LT 90
FD 10
LT 90
FD 10
```

"I know I've got to keep doing this so ... ". She cleared the screen, went into the teach mode, and began typing the WALL procedure shown to the right. After entering the above section, Sandy saw even more clearly the pattern that she was using, typed an END statement and went into the editor to put all but the first line of the procedure into a REPEAT statement. All of the above was done without making a note and with exact recall of the immediate mode work; eliminating the double FORWARD and the lateralization error, as it was entered in procedural mode.

The work of Carol was done mostly in procedural mode and immediate mode was used only to test for a larger sized circle to put into the procedure instead of the small one.

REPEAT 72 [FD 4 RT 5] instead of REPEAT 72 [FD 2 RT 5]

Mark used immediate mode when he was unsure of the forward length required.

ST FD 50 BK 20 BK 20 FD 5 FD 5

Then he calculated the total forward distance. Later he used immediate mode when he received an error message following the execution of a procedure which started with this line:

```
SET X,Y (-100), (-100)
```

He first tried to debug this line in the editor by removing the space between SET and X. After working in immediate mode he returned to the editor and entered:

```
SETY (-100)
SETX (-100)
```

Observe and Change

As was mentioned in the description of this strategy, observe and change was used most often in the case of lateralization bugs and PENUP and/or PENDOWN bugs. Many of the lateralization bugs related to left and right turns of 90 degrees. The following sample from the immediate mode work of Barry, shows the lateralization bug with a 45 degree angle and in each case he would observe and make the necessary change.

```
FD 90
RT 45
FD 20
FD 20
LT 45*
RT 90
FD 40
FD 20
FD 20
LT 45*
RT 45
RT 45
```

Later in that same observation while in procedural mode in the editor, Barry typed:

```
PU
HOME
REPEAT 4 [FD 90 LT 45]
```

When he tested the procedure he said, "Oh, I forgot to put PENDOWN." Observing immediately that he had omitted PD, he reentered the editor and added PD after HOME.

Trevor's work in procedural mode shows similar examples of the observe and change strategy of debugging. The end of his FLOWER procedure was:

```
PU SETPOS [-30 -30]
REPEAT 90 [FD 2 RT 1]
```

Upon testing he commented, "I forgot to put PENDOWN." He reentered the editor and added PD. He then tested again and this time was able to see the turtle draw the arc stem he desired. His comment was, "Oh, oh! I forgot one thing. Before I put RT 180 (his immediate mode work) and it went this way. Now it went through the flower." Following this observation he edited again with his procedure then looking like this:

```
PU SETPOS [-30 -30] PD
RT 180 REPEAT 90 [FD 2 RT 1]
```

Guess and Check

The uses of guess and check were many and its application varied from a very random guess to a very educated, precise guess.

Connie, in making a rectangle, had the following procedure:

```
TO YOU
REPEAT 4 [ FD 60 RT 90 FD 90 RT 90]
END
```

When she tested this procedure she commented, "It goes over twice so I should change that (meaning the number of REPEAT's) to 2." She

pauses. "Or maybe only once." She edited the procedure to read REPEAT 1 [FD ... and tested again. "It only went halfway - have to have 2," she said as she edited it to REPEAT 2, and then tested again to be sure.

Don's work illustrates a fairly random series of guess and check strategies. He was trying to locate one circle within another, and at no time showed any consideration of the radii of the circles. The following sequence is taken from the observation notes.

```
. . .
HOME
LT 45
PU
FD 45
PD
```

Don tested and said, "I should have put PENUP before HOME." He entered the editor, added PU and changed LT 45 to 59, commenting, "When I put 45 it didn't turn enough. He tested again, then entered the editor and added LT 85, and then tested again. He said, "I'm trying to find the right spot," and entered the editor again. The resulting procedure was then added to:

```
. . .
PU
HOME
LT 59
FD 45
PD
LT 85

PU
FD 45
```

He tested again and added:

```
RT 35
```

He tested again and added:

REPEAT 30 [LT 23 FD 13]

At this point Don tested again, observed the missing PD, and commented, "I didn't put PENDOWN. I've got to put it (meaning the small circle) up a bit more." He then reentered the editor, made a changes, and tested three more times in the following sequence:

Deleted RT 35 and put in PD
Changed the last FD 45 to FD 49
Changed the FD 49 to FD 57.

Although Don was content to stop at this point, his circle was not symmetrically located to the previous one. His guess and check method was not the most desirable and one would likely recommend that he follow the pattern found in the earlier part of the procedure.

In trying to adjust the location of the teeth on the face, Cindy changes the FD 35 to FD 45, then to FD 55, then to 60 and finally to 65, to get the precise location desired. Although her first guesses could have been more accurate, she had little choice for strategy except to guess and check. This is also the case with Mike. He changed from RT 125 to RT 135 and then to RT 140 while determining the angle for the letter V. He later worked to get the precise turn desired by adjusting from LT 140 to LT 100, to LT 105, and finally to LT 103.5.

Playing Computer or Playing Turtle

Playing computer or playing turtle as it is often called in Logo literature, can include a variety of approaches. It was used by students who were working in immediate mode as well as procedural mode.

During the second observation with Cindy, she was observed moving her upper body to help determine the desired turn. During the third observation, while adding to a procedure, she said, "I've got to stand up." She took two paces, turned left, and then returned to the chair and entered a few commands. Later, while still having difficulty with the procedure, she took some paper, got up, walked, talking aloud and recording as she went. She then entered commands from her notes into her procedure. This was followed by four other notations of Cindy turning her body or a part thereof to help her determine the route the turtle should take. On two occasions in this same observation Cindy went line by line through the listing of the procedure, matching it to the diagram.

A similar situation existed in the second observation of Murray, who was designing the letters B E M I N E. He began his debugging session by comparing his notes to the listing of the procedure in the editor. When this did not reveal the bug he went through the listing line by line, drawing a diagram. This was successful in finding one bug. He then continued through the listing, drawing as he went. He responded with "Ah hu. Right there there should be a FORWARD." Later, while designing the next step, he looked at the graphic on the screen, his hand traced the moves, while his head and shoulders followed the turtle.

Clear Screen and Start Again

CLEARSCREEN (CS) or DRAW was used by a number of students and on a few occasions was used repeatedly before a debugged program was achieved.

The following example comes from the work of Linda. She was trying to locate the turtle at an appropriate starting point before beginning her lettering.

```
PC 1
LT 90
RT 60
FD 50
DRAW
PU
LT 65
FD 100
FD 50
PD
RT 90
FD 20
DRAW
```

Trevor's work was actually a continuation of the task from the previous day on tessellating hexagons. Since he had left the session with bugs in his work he chose not to load his files, but rather make a fresh start. He redefined the shape, an irregular hexagon, and then in immediate mode used the procedure with other commands to discover a way to tessellate them.

```
LT 90 SHAPE   CS
SHAPE LT 180 SHAPE   CS
SHAPE LT 180 RT 45 SHAPE   CS
SHAPE RT 180 LT 180 LT 180 LT 90 SHAPE   CS
```

"Not quite!"

```
SHAPE RT 45 FD 20 RT 180 LT 90 RT 180 LT 45 SHAPE    CS
```

After 16 tries Trevor had a procedure that linked two shapes together in the manner he wished. It looked like this:

```
TO T
SHAPE
RT 45 FD 20 RT 90
SHAPE
END
```

Six more tries in immediate mode with CS between each to get a fresh start resulted in a procedure that would link a third SHAPE to the two already joined.

```
TO S
T
BK 20
RT 45
RT 45
RT 45
SHAPE
END
```

Delete and Start Again

Although this is very similar to the above category, it refers to work done in the procedural mode and is therefore treated separately.

While making his tunnel, Barry had completed part and responded, "There! That's how I want it." Following this he added:

```
BK 25
RT 45
FD 50
LT 45
FD 25
LT 45
FD 25
```

He tested but was not pleased with the results so he deleted all but the first line. Then he entered:

```
BK 25
-----
LT 45
FD 25
RT 45
FD 25
RT 45
FD 25
```

During an observation with Cindy, a procedure for MOUTH (on the left below) was written to add to the exixting face procedure.

```
TO MOUTH
PU
FD 20
PD
LT 90
FD 30
RT 90
FD 10
```

```
TO MOUTH
PU
FD 35
LT 90
PD
FD 20
RT 90
FD 5
LT 90 RT 90
RT 90
FD 50
RT 90
FD 5
RT 90
FD 30
END
```

After testing MOUTH, the superprocedure was brought into the editor and MOUTH was added after NOSE and then tested again. The result was not as desired so MOUTH was edited and the first FORWARD changed to FD 90 and retested. Next the LT 90 was removed. Following testing a RT 90 was added in its place and again tested. At this point all of MOUTH was deleted and a fresh start made. The result was a long narrow rectangle shape made with the MOUTH procedure on the right above.

Simplify

Because of uncertainty with the syntax of the SETPOS command, some students were observed simplifying their approach. Vicki was observed entering the following line:

```
SETX 50 100
```

When she tested it she received the message, "I don't know what to do with 100." Vicki responded with, "Oh, I did it the wrong way." She then deleted the line and entered:

```
PU
LT 90
FD 60
```

Her comment was, "This is easier." When beginning her session the next day, Vicki was observed looking around the room at the wall charts. When asked what she was doing, she said she was looking for a chart that told how to set X and Y, but couldn't find one. Therefore she proceeded to use the same approach she had used the day before.

During an observation Sandy edited a procedure from the previous day in which she had correctly used SETPOS. She made the following change:

```
SETPOS [0 -80]  to  SETX -100
                    SETY -50
```

She commented that she was not sure of SETPOS so was going to use the other. Mark had handled a problem with SETXY in Terrapin Logo in a similar way, by using SETX and SETY when he was unsure of the syntax of the SETXY command.

Using the REPEAT command presented problems for a number of students. While making a rectangle, Cindy removed the REPEAT 4 which was giving her a square, and entered the eight commands instead. (Barry retained the REPEAT command but solved his bug by switching the task to a square instead of a rectangle.) REPEATs were removed by both Ted and Sandy while making hexagons.

Trevor tried to tessellate a hexagon which he called OBJECT1. In attempting to get rid of a bug he decreased the number of REPEATs from six to two and then removed REPEAT completely .

```
TO ROW
REPEAT 6 [OBJECT1 20 RT 180 FD 20 OBJECT1 20 LT 45 FD 20 LT
          90 FD 20 LT 45 RT 180]
END
```

```
TO ROW
REPEAT 2 [OBJECT1 20 RT 180 FD 20 OBJECT1 20 LT 45 FD 20 LT
          90 FD 20 LT 45 RT 180]
END

TO ROW
OBJECT1 20 RT 180 FD 20 OBJECT1 20 LT 45 FD 20 LT 90 FD 20
    LT 45 RT 180
END.
```

Later in the same session he again removed the REPEAT from another procedure, worked with the contents in immediate mode for a time, and then reinserted the REPEAT.

```
TO ROW.L
REPEAT 6 [ROW ROW LT 45 FD 20 LT 90 FD 20 LT 45 RT 180 BK
          20]
END
```

This is the only observation of a student simplifying and then returning to the more complex. Although this did make a nice design that Trevor was pleased with, it still was not bug free with respect to the task he was trying to accomplish.

On a number of occasions students using the strategy of simplifying were observed avoiding the use of subprocedures as they found them too complex. One such example appeared in the work of David. He had a procedure named HEAD and then began:

```
TO EYES
RT 60
<CONTROL> C
```

David said, "I don't know how. I'm just going to put it all together." He then entered the HEAD procedure in the editor and proceeded to add the eyes onto the part already contained in HEAD.

Shauna removed a subprocedure from the end of the first REPEAT statement and added it later in order to simplify the action and in the end had the following procedures:

```
TO ROW                          TO F
REPEAT 1 [SHAPE ALL D]          START
RT 90                           ROW
FD 20                           ROW
RT 45                           START
FD 20                           PU
RT 45                           BK 30
FD 20                           PD
LT 45                           ROW
REPEAT 1 [E]                    ROW
S                               END
END
```

She had retained the REPEAT statements with the hope that she could increase the value from one back to eight when she got the bugs worked out. This did not happen, and neither did the deletion of the unnecessary commands.

Ray simplified his procedure by removing a subprocedure from the calling procedure, tested the procedures up to the offending subprocedure to be sure they were fine. He then debugged the subprocedure.

Vicki's simplification showed her uncertainty with using variables. She began her procedure:

```
TO RECTANGLES
FD :SIDE
```

Then she changed it to FD 50 and continued without using variables.

Seeks Information

A variety of information seeking strategies were observed. Cindy consulted a book to find out how to make a rectangle using the REPEAT

command. She could not locate it. Vicki looked around the room for a chart on the use of SETPOS or SETX and SETY but was unable to find one. Barry received help from his teacher on one occasion when he was using the shifted bracket ([) on the Apple //e, in a procedure and could not decipher what was wrong. Shauna consulted with a friend as to why her procedure was not working. Her friend looked at it briefly and said that she had the pen up. She typed PD and tried it again with the same result. The friend said, "It's in your procedure." and walked off.

On many occasions questions were asked of the researcher. These were treated, if possible, as a case of the student thinking aloud, or asking a retorical question. When this was not satisfactory with the student, the researcher would respond with, "What do you think?" or "Do what you think is best!" There were a total of eleven instances where the researcher did give information. This did not include assistance given by the researcher in disk and file handling and assistance with the editor commands and procedures. These cases were not recorded as they were considered not to interfere with the data collection. The following examples show instances where information was given by the researcher.

An error message resulted from Cindy's work.

```
TO ME
PU
LT 90
FD 24
PD
REPEAT 90 [ FD 30 RT 30]
* TO JA  (a subprocedure)
PD
REPEAT . . .
```

Cindy was unable to find the bug on her own. After waiting for a while to see what she would do, assistance was given by the researcher. Her own teacher, present in the room, was busy with the class. Later in the same session, while working with circles, assistance was given by the researcher, to determine the number of REPEATs so that control could be established over the stopping point on the circle. Cindy could make a circle but had little idea of how to get it to stop when the circumference was complete. On a later occasion Cindy was trying to carry out a move, had turned to the right, gone forward, and then proceeded with the next part. She was perplexed as to why this did not work, so the researcher drew a diagram for her of the turtle's movement and heading. She was then able to fix her procedure.

David had several REPEAT statements which appeared to be a random collection of commands. On one occasion the resesarcher talked through the commands in an attempt to explain what would happen, and asked if that was what he wanted.

A subprocedure that was causing difficulty for Sandy was simplified when the researcher suggested moving the turn command from the beginning of the REPEAT to the end so that the subprocedure could be more easily joined to itself.

```
TO P :SIZE
REPEAT 2 [RT 45 FD :SIZE RT 45 FD :SIZE RT 90 FD :SIZE]
END

TO P :SIZE
REPEAT 2 [FD :SIZE RT 45 FD :SIZE RT 90 FD :SIZE RT 45]
END
```

When Mike ended up with a recursive procedure that he could not control, the researcher talked with him about what he wanted it to do. Since he was familiar with superprocedures, it was suggested that the writing of a superprocedure which called for his subprocedure the appropriate number of times was one solution, and then he removed the recursive line.

During Linda's first observation session she became totally confused and flustered. Assistance was provided for her, as it was felt necessary for her self-esteem and to establish rapport for later sessions. With relief she said, "Thank you for helping me."

Following a Pattern

For those students doing an assignment on symmetry, one would expect to see extensive use of this strategy. This was the case in the work of Barry and Connie. Cindy also used this strategy in working out the details of her face project.

Ray was able to follow a pattern to debug the DOT command statements which had bugs in them. He had found the location of the first DOT by using the PRINT POS command and then had written a succession of statements. These all included negative numbers, and since the first was to go at the location, DOT [-120 -13], he in error decreased the X value to move to the right.

```
DOT [-130 -13]
DOT [-140 -13]
```

Ray quickly saw the pattern, changed the values, and added the rest of the DOT commands he desired.

In establishing the origin point for each row in his triangle tessellation, Ray was able to see a pattern and use it. He said, "I do FD 30, LT 90 and then 50 less than the one before. Then RT 90 and FD 15." This was a fairly complex pattern and was established quickly. In actual fact he was not subtracting 50, but was using 50 and then making adjustments.

Sandy was quick to see patterns, both when building the castle and tessellating hexagons, and as a result included the necessary lines in a REPEAT statement. Although the example here was used in building the procedure, it was useful later in debugging a part of the procedure.

```
TO WALL
FD 100
RT 90
FD 10
RT 90
FD 10
LT 90
FD 10
LT 90
FD 10
```

She saw the pattern and changed the procedure.

```
TO WALL
FD 100
REPEAT 10 [RT 90 FD 10 RT 90 FD 10 LT 90 FD 10 LT 90 FD 10]
. . .
```

Error Messages

Recorded here are some of the bugs which resulted in error messages being printed by the computer, and how these were used to remove the bugs.

The following is from the work of Sandy.

```
TO WALL
FD 100
REPEAT 10 []
RT 90
FD 10 . . .
```

The contents of the REPEAT, which had been written first as a string of commands, appeared following the REPEAT instead of within the brackets. When she received the message, "NOT ENOUGH INPUTS TO REPEAT", this was quickly corrected.

Sandy received an error message as a result of this procedure:

```
TO P :SIZE
REPEAT 2 [RT 45 FD :SIZERT 45 FD :SIZE RT 90 FD :SIZE]
END
```

The necessary space was inserted between :SIZE and RT after the message, "SIZERT HAS NO VALUE IN P" was read.

Shauna and Vicki received error messages saying, "I DON'T KNOW HOW TO REPAET", when they had misspelled REPEAT. REPAET was corrected to REPEAT.

Troy's work resulted in error messages because of the variable inputs.

```
TO REC2 :LENGTH
FD :LENGTH
RT 90
FD :LENGTH
```

He tested this by entering REC2 67 89, and received a message that the computer did not know what to do with 89. After several attempts to debug this, which included changing the variable names, Troy realized that he needed two variables in the name and then typed:

```
TO REC2 :SIZE :WIDTH
FD :SIZE
RT 90
FD WIDTH
```

This resulted in the message, "I DON'T KNOW HOW TO WIDTH: JUST BEFORE LEAVING REC2". Troy responded with, "Oh, I've got to use a variable - dots." He then tested the procedure with REC2 40 80 40 80 and another series of messages and debugging attempts followed. After many attempts, often ignoring the messages, Troy left this bug unsolved. When using the procedure as a subprocedure the next day he was again faced with the bug and did eventually conquer it. Troy received more error messages than any other student. He had considerable difficulty handling them and on several occasions tried to ignore them. He also was the receiver of the message, "LOGO SYSTEM BUG", which could not be removed, and resulted in the loss of all the procedures he had typed in that day, as it would not allow him to SAVE or continue work. His work was reentered on another machine from the notes of the resesarcher. No one was aware of what had caused the bug or how to get rid of it.

Barry also received error messages.

```
TO 17
REPEAT 4 [ . . .
```

Barry received a message that TO doesn't like 17 as input so he changed the name to JOE. He later received a message saying there were not enough inputs to FORWARD, as a result of the following statement.

```
REPEAT 4 [FD]
```

The necessary input was added and Barry continued work. Barry also received the error message, "THERE IS NO PROCEDURE NAMED 4[FD90".

Barry had two bugs here. The space was added but the hidden bug, the shifted bracket, was resolved after consultation with his teacher.

Other error messages resulted from spacing bugs, unmatched right brackets, and extra letters.

Changing the Plan of Attack

When having difficulty with the placement of a section of the symmetrical design, Carol switched the location of the axis of symmetry to make the removal of her bug easier.

Mark's work is an example of several students, who when having difficulty placing the turtle, would change their approach by typing HOME and then attempt the required location again.

```
. . .
RT 90
PU
FD 45
RT 90
PU
HOME
. . .
```

Mike's change in plans was made to get rid of the limiting effect of having HOME in the procedure. It would not allow him to place the procedural drawing on the screen in a variety of places. He was designing a heart by making the upper right portion, returning HOME, making the upper left portion, and then adding the bottom part.

```
TO H
REPEAT 115 [RT 2 FD 1]
PC 0
HOME
PC 3
REPEAT 115 [LT 2 FD 1]
FD 60
LT 78.5
FD 60
REPEAT 5 [RT 35 H]
END
```

When Mike tested this procedure he commented, "See, it goes to there and then goes to HOME. I think I can fix it." He made all the necessary changes without a single slip. This was an exciting example of very clear, logical thinking.

```
TO H
REPEAT 115 [RT 2 FD 1]
FD 60
RT 78.5
FD 60
REPEAT 115 [RT 2 FD 1]
REPEAT 5 [RT 35 H]
END
```

The last line still contained a bug as it has a recursive call within a REPEAT command, but the task he had set out to accomplish was well executed.

Abandon or Change Task

Switching to a square rather than a rectangle was done on more than one occasion. As mentioned earlier, Barry did this.

When working on his Easter project, Mark changed the task from making several eggs to making one egg and decorating it. The reason for this was a fixed SETY command in his procedure, which did not allow the egg to be placed at various locations on the screen.

Linda, as well as other students, changed the task of writing L O V E to L O U E in order to avoid the difficulty of the diagonal line.

David switched from round eyes to V shapes because of the difficulties he had in making small circles.

When the nose on the face Barry was making ended up in the eye, he changed his plan, made a corresponding part for the other eye, and

then began again on the nose. When difficulty arose with the arcs in his symmetry project, Barry decided to add squares instead of arcs.

Vicki decided to make a diamond when the triangle procedure resulted in a bug. When starting this session, she stated that she was not going to complete the task begun earlier as it was too hard. Only two other instances of abandoning the task were recorded, and one of those was by Barry who wished to experiment with something else. He said of the abandoned task, "Will work on him tomorrow."

Several other cases of unfinished projects were observed but not recorded as such, because the teacher had asked the students to work on something else. Thus the task could not be considered as abandoned by the student, and in fact the one mentioned above by Barry may not have been permanently abandoned either. The observation in which this happened was the last one with him.

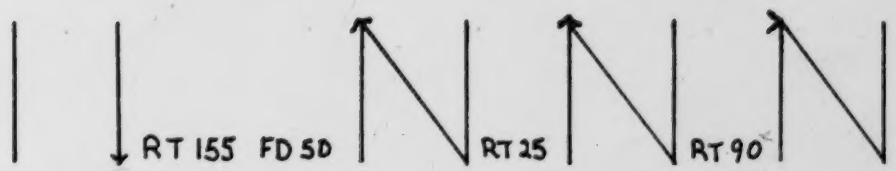
Other Strategies

Mathematical Calculations. The following are fairly typical of the instances where mathematical calculations were recorded. Carol had made a square with sides of 120 turtle steps, and then wished to center a circle within it. With her finger on the side of the square she said, "That's FORWARD 120. I want to go halfway (moves finger up halfway), so that's 60." In making the large letter N Murray had made the first stroke of the N and then the final stroke, leaving the diagonal to be done last. With the turtle at the bottom, heading 180, he typed RT 155 FD 50 and the N was complete, but he needed to reposition the turtle for the next letter so some calculation was necessary. Considering the RT 155 and using his hand, Murray said,

"Turn RT 25 to face up. Then 90. That's RT 115 to do the whole thing because I want to face that way (pointing to the right)". Figure 12 depicts the steps of Murray's work.

Figure 12

Murray's Calculations



Calculations were not always correct. For example, Ray, following a series of turns to both the left and right (RT 180 LT 50 RT 20), added or subtracted incorrectly and had to work in immediate mode some more to come up with the RT 150 that was required.

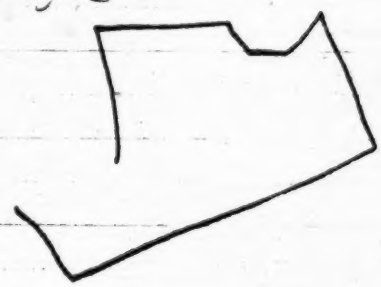
Drawing Diagrams. Although few diagrams were drawn, they were an effective means of debugging. Laura's diagrams (Figure 13) were done in three stages as she debugged her Valentine Card procedure, while Murray's small diagram was carried only far enough to work out the missing commands. While Connie had been working, the observer sketched a diagram of her project. When she had difficulty debugging

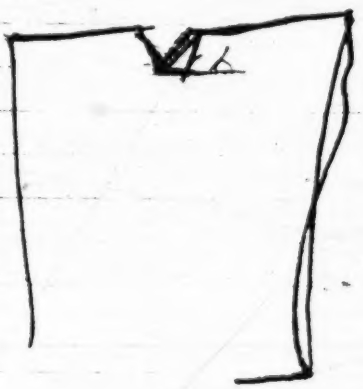
Figure 13

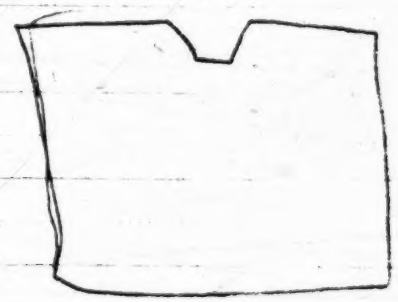
Laura's Notes and Diagrams

Valentine Card.

PU
RT 90
FD 70
RT 90
PD
FD 80
RT 90
FD 30
RT 45
FD 10
LT 45
FD 10
FD 40
FD 30
RT 90
FD 120
[illegible]
RT 90
FD 82
RT 90



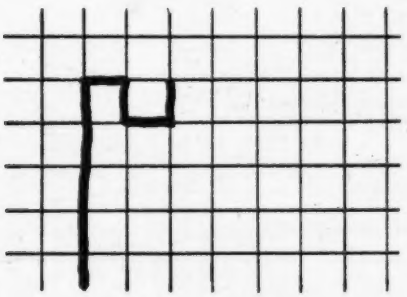




she used the observer's pencil and paper and made a drawing which she retraced several times. Sandy's diagram, shown in Figure 14, was done on graph paper, which was very effective for her as she had a series of short lines and turns. Don's comment was, "I just picture it in my head." This was likely the case with all students at some time, but those students who used pencil and paper diagrams found them profitable.

Figure 14

Sandy's Graph Paper Drawing



Note Taking. Note taking was a much more common strategy than the drawing of diagrams. As was mentioned in the discussion of the immediate mode strategy, which often accompanied note taking, students faced difficulties with their notes, as well as using them as a valuable tool. Several students, including Laura (Figure 13), Lana, Linda, and Murray, not only made notes, but kept a notebook. Others,like Cindy, made notes on scrap paper and kept them only as long as they were needed to debug the program. She made notes only during some of the observation sessions. Her notes were usually short sections which were then transferred to the current procedure. Most

of her notes were made while she walked or moved her body or hand, playing turtle. Some were made while observing the screen display and some were actually copied from the editor. Murray made notes along with the immediate mode work, but more often while just planning, looking at the screen, and accompanying his physical movements.

Abandoned Procedures. A number or procedures that didn't work or were no longer needed, appear on the student's disk files. Troy started his session with the following procedure.

```
TO REC
REPEAT 2 [FD 200 RT 90 FD 160]
END
```

Upon testing it he commented, "That didn't work. Oh, I know how to do it." He proceeded to write REC2 which used variable inputs, and REC never was touched again. Vicki wrote a procedure called RECTANGLE that did in fact make a rectange, but then wrote another called REC, which was a different size. She had attempted to make the first with inputs for size but was unsure and so when she needed a rectangle of another size she did not consider the first one or attempt to edit the length values. Vicki also wrote a procedure for a triangle which she was not able to debug and so it was abandoned but left in the workspace. Shauna wrote a variable input rectangle procedure which worked perfectly. She then abandoned it and wrote two more fixed value rectangle procedures. When she was in the process of writing the third identical move procedure, named E (G and L are the same) she realized that it was unnecessary, did not finish it, nor did she use L again. No attempt was made to clean the workspace. However, clean workspaces were kept by a number of students who would erase those

procedures which resulted from a typing error. Such was the case when Mark typed CUPIT instead of CUPID, the name of his procedure.

Change Order of Procedures. Ray found it necessary to change the order in which he called the subprocedures on two occasions. He, along with others, was noted chaining procedures together so that the procedure currently under construction would call the previously completed one, which in turn called the one that was completed before it. This was usually an effective strategy. One time, however, he had the previous procedure call the new one and instead of adding to the top as before it was now necessary to add to the bottom and call two procedures. A more common cause for removing a procedure was due to the use of fixed positions within it. On a number of occasions a MOVE procedure had to be removed from another procedure so the procedure could be used repeatedly without reusing the MOVE procedure.

Using <CONTROL> G. Shauna made three attempts at using <CONTROL> G to locate the point at which her tessellation procedure went astray. She found it difficult to stop it at the right moment and then appeared to not be sure how to use the information given in the message to her advantage. She then deleted the whole procedure, worked in immediate mode, and then rewrote the procedure. While working with the mouth on the face, Cindy asked, "Is there any way you can pause it when it's doing this?" Such a strategy would have been helpful to her.

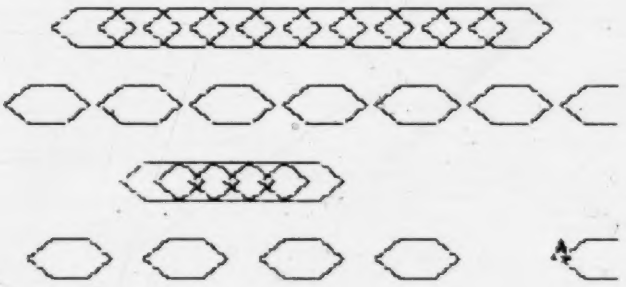
Using Variables. Variables were used in procedures as a means of debugging when the length or the number of REPEATs was not known. Ted's work (Figure 15) illustrates a combination of these two uses.

He was attempting to tessellate irregular hexagons (which he called PENT), spacing them end to end and trying to make enough to fill the width of the screen.

Figure 15

Ted's Variable Input Procedure

```
TO SETUP :P :C
REPEAT :P [LT 90 PU FD :C RT 90 PD PENT]
END
```



Using PRINT POS. Ray's repeated use of PRINT POS as a debugging strategy worked effectively. On one occasion, after making a number of immediate mode moves to position the turtle correctly for the first row in the hexagon tessellation task (Figure 4), an appropriate location was found. He then worked in immediate mode to find the correct location to start the next row. Then Ray typed PR POS, mentally noted the values he got back (-95 -55), edited the procedure called ALL, and entered the values into the SETPOS command.

```
TO ALL
MOVE
REPEAT 5 [S MOVE2]
PU SETPOS [-95 -55] PD
REPEAT 4 [S MOVE2]
```

Experimenting. A number of instances of experimenting were noted. Some of these consumed the whole period or a fairly large block of time. Although these were left as unclassified blocks when the student indicated that he just wanted to experiment, the researcher is aware that in fact debugging was in some instances taking place and the experimenting could be called a strategy. It would be very much like some of the guess and test sessions that took place. Some of the experimenting had been assigned by the teacher as indicated earlier, while some was at the wim of the student. Carol wanted time out to experiment with circles, while a number took some time to experiment with color. Marks comment about color was, "It's random - can never tell how it turns out."

Multiple Strategies

Although each strategy type has been discussed separately and examples have been cited to illustrate the debugging strategies, it is important to realize that often multiple strategies were used as a joint attack on a particular bug.

Three strategies often seen together were immediate mode, physical referent (ususally finger or hand), and guess and check. Searching through the listing line by line and using a physical referent were also often used together. Some of these combinations are discussed further in the section on how strategies relate to bugs.

The following sample comes from the work of Laura, called Valentine Card. A copy of her notes and diagrams appeared in Figure 13. She worked in immediate mode, taking notes in a notebook.

Types	Writes	Comments
PU		
RT 90		
LT 180	LT 90	
FD 50		
FD 20	FD 70	
RT 90	RT 90	
PD	PD	
FD 50		
FD 30		
BK 80		
FD 80	FD 80	
RT 90	RT 90	
FD 30	FD 30	
RT 40	RT 40	
FD 10	FD 10	
LT 40	LT 40	
FD 10	FD 10	
LT 40	FD 40	
FD 10*		
RT 40*		
FD 30	FD 30	
RT 90	RT 90	
FD 80		
RT 90		"I'm going to make it longer."
LT 90		
FD 20		
FD 20	FD 120	"90, 100, 120."
RT 90	RT 90	
FD 80	FD 80	(This was later changed to 82.)
RT 90	RT 90	
FD 40		"Oh, I never went far enough."

At this point Laura decided to go into the editor and enter what she had recorded in her notes and then work on the length bug that she was aware of in the last line above. She typed quickly with the left hand, her right hand on the book. The first line, LT 90, was entered as LT 9 0, which resulted in an error message. Laura quickly fixed this. While entering the lines above she skipped two lines marked with asterisks and as a result had another bug to fix. Her response was, "Oh, yuck. It didn't work." Using the cursor as a line marker at the front of each line and moving the cursor with <CONTROL> N,

Laura played turtle with her right hand moving through the air. When the cursor arrived on the first LT 40 line, Tracy changed it to RT 40, left the editor and tested the procedure. She laughed and said, "That wasn't the problem." She once again entered the editor and moved through the listing, this time using her pencil to draw the turtle's path on paper. She changed the RT 40 back to LT 40 and then removed the next LT 40 and tested. She reentered the editor, and instead of adding in RT 40 where the LT 40 had been removed, she typed FD 40. She again tested and then reentered the editor, adding in a LT 40, and again testing. Laura and the listing were at this stage becoming confused. She entered the editor once more, and using pencil and paper she drew and followed the listing until she came to the first missing statement. From here she ignored everything below, inserting new lines as she drew. She was typing with the left hand and drawing with the right. After entering the new lines she then deleted everything below and tested again. Laura really didn't use her notes to debug this section and in fact the omission also occurred in her notes so they would not have been helpful if she had referred to them. An error, RT 90 instead of RT 40, still existed at the end of the observation.

In this section of Laura's work we see her extensive use of immediate mode, use of notes, (albeit not too accurately), guess and check, observe and change, using error messages, playing turtle, drawing a diagram; and deleting a section.

APPENDIX 10

Observation and Log Notes Illustrating Student Interest

Observation Notes on Cindy

Cindy was an example of a very exuberant worker. She came for her sessions with a definite task in mind and enjoyed explaining what she was about to do. She expressed her pleasure throughout her work with such expressions as, "Oh, I did it. Goody!" (Claps hands.) "I did it!" "I got it perfectly!" "Perfect! Now make little ears." "Did it! Now little teeth." Many of her comments are recorded here to show the sort of monologue that took place as she was busy debugging her program.

"His eye is too big? How do I shorten it down?"

"Oh, I did it. Goody!" (Claps hands.)

"Now I can change this one."

"Now I'm going to put them (subprocedures) all together."

"Oh, yuck!"

"From there (points) I've got to get over here. I'm going to shorten that circle."

"I did it!"

"Now I'll look at the other one." (Model or pattern) "Repeat 12, that's what I did."

Cindy not only chatted about her work, but used a fair amount of physical movement (playing turtle) to work out her turns and moves. She also worked on a very similar program at home on a different make of computer. She brought computer books to show the researcher and

talked about working on the computer with her mother. She was anxious to learn more.

The following notes are taken from the January 28 observation of Cindy's work. Although she faced many bugs she was cheerful and exurberant throughout.

Cindy consults a book from the library (Learning With Logo by Dan Watt) to get help on making a rectangle.

"I've got to stand up." Cindy takes two paces and turns left, then sits again.

Cindy takes paper, gets up, walks, and records.

Cindy wants a rectangle but doesn't get it with REPEAT 4, which gives her a square. She deletes the REPEAT 4, starts in immediate mode, and then gets up and turns, types, turns in her chair, types, moves upper body, types, moves upper body, and types. She then types <CONTROL> T so she can see the text screen, and copies down the commands. She then enters the commands into the procedure. When completed she says, "Perfect! Now make little ears. The next program will be called ears - EAR1 and EARS2."

Observation and Log Notes on Mark

Mark worked quickly, always ready with the next step. He was at times a difficult programmer to follow as he would jump from one subprocedure to another as he was debugging. As he worked on his Valentine's project he made an arrow through the heart and began the lettering. The following comments were recorded as he worked.

"I'm going to correct that turn."

"I'm going to take the pen and lift it up."

"Now I've got the arrow. Now I'm going to extend it, PENUP, come over here and make it sort of look like it went into the heart."

"Let's see if this works. (Tests.) It worked." (Very matter of fact.)

"What I'm going to do is after I make this arrow I'm going to print down here, (Points) WILL YOU BE MY VALENTINE."

"Oh, no!"

"That's more like it."

"I think I did this right."

"That should set that there. What is wrong?"

"I forgot PENUP. That's when you lift the pen and if you don't it leaves this streak behind."

"One thing I learned is SAVE it after you get a few things done because I was typing in this one program about three pages long and the power shorted and I lost the whole thing." (He saves.)

"Full screen adds the extra for lettering. My turtle is down here so you won't be able to see it. Now W. I hate doing lettering." Although he said he hated doing lettering, he did not show that attitude, but worked along in his cheerful, chatty way. He chose to do 20 letters!

The following comments were made in the log notes of the researcher about Mark. They give some insight into the approach and attitude with which Mark works.

January 24, 1985

Mark works quickly and with confidence. Mark's difficulty with SETX, Y (-100), (-100) was worked out with little sign of frustration. Charts or a book would have been a useful reference.

January 31, 1985

Mark truly does work quickly -- switching from one procedure to another -- and back -- sometimes making adjustments to the superprocedure and sometimes to the subprocedures. He was so flitty and I was getting tired (last student before lunch) that I really had trouble following. He too said he was confused, but he doesn't take time to stop and plan -- an idea flashes and he operates on impulse. He also makes changes in more than one place at a time and then when he tests it is hard to tell which change did what. Part way through he changed his plan, deleted a whole section and then started on LOVE.

February 01, 1985

What do I say? Between jumping from one procedure to another, editing without testing, making multiple changes, and the quickness of Mark's work, it is very difficult to follow. I think that he could be fantastic if he was not so undisciplined in his approach.

February 07, 1985

Confident Mark came in ready to tackle a St. Patrick's design today. Others are still doing Valentine's but he has finished! He chose to do a shamrock.

He was his usual quick moving self. He spent a fair bit of time juggling the three circles and was aware when he finished that it was not balanced, but it was "good enough". His work certainly is not

what you would call carefully thought out or well preplanned but rather built on impulse. He is never stuck for an idea, though.

After making the second circle he had difficulty locating a suitable starting location for the third circle. He finally abandoned his moves, put the pen up, went HOME and started out from there. (At no time did he go back and delete the four unnecessary lines.)

After completing the third circle he decided to adjust the stem to his uneven leaf but in doing so he left it open at the bottom. His stem is now more strangely made. He was pleased to have done a whole project in one period and guesses he will have to "do Easter now".

February 08, 1985

Well, it was on to Easter! Mark decided to make an egg. He experimented to make a half circle, then partitioned the half circle and asked me if I knew how to make an egg. Following my negative responses he set to work experimenting. He sure is a good logical thinker -- not always organized -- and keeps a number of things in mind. He will experiment in immediate mode, go into the editor and transfer his results, never using pencil and paper and often making further refinements.

His attempt to make eggs was abandoned after one try when he realized the SETY he had used in EGG spoiled EGGS. Instead he chose to decorate his egg and started on a DEC procedure.

Mark has his own disk. Sometimes he saves his work on the class disks, but those procedures he considers special he puts only on his disk. "Other kids look at your programs on the class disks and sometimes they copy them and call them theirs."